THREE-DIMENSIONAL OBJECT RECOGNITION THROUGH THE INTEGRATION OF EDGE AND SURFACE ORIENTATION MAPS

By

HONGYOUNG AHN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1991

ACKNOWLEDGEMENTS

The author would like to express his deep gratitude to his adviser and supervisory committee chairman Dr. Julius T. Tou for his invaluable inspiration, encouragement, and advice during the course of this work. Without his guidance and support, this work could never have been done. He would also like to thank his committee members, Dr. John Staudhammer, Dr. Leon W. Couch, Dr. Jose C. Principe, and Dr. Yuan-Chieh Chow, for their valuable discussions and help.

Appreciation is owed to Russell Walters for his editorial help and to his colleagues at the Center for Information Research for their kindness and encouragement. Special thanks goes to Mr. Jong-seok Park for stimulating discussions and helping in setting up the experiment.

Finally, he would like to thank his wife Jeong-Ja, his son Jae-Hyun, and his daughter Eun-Kyung for their sacrifice, constant encouragement, and understanding.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

THREE-DIMENSIONAL OBJECT RECOGNITION THROUGH THE INTEGRATION OF EDGE AND SURFACE ORIENTATION MAPS

By

Hongyoung Ahn

August 1991

Chairman: Dr. Julius T. Tou
Major Department: Electrical Engineering

This dissertation presents a model driven approach to 3-D object recognition using a surface based, view independent object representation. The primitives of the model are elementary surfaces of homogeneous curvature characteristics. A surface orientation map generated by a photometric stereo method and an intensity image are used as input.

Segmentation of an image is the most difficult and crucial step toward symbolic transformation of a raw image, which is essential in object recognition. An integrated approach is taken to reliably segment the surface orientation map. In one channel, edges are extracted and linked to produce edge contours from an intensity image. Edge contours are then segmented based on the curvature values along the contour. The segmented edge contours provide initial region boundaries. In the other channel, Gaussian and mean curvatures

are estimated at each pixel from the least squares local surface fit of the surface orientation map. A labeled surface type image is then generated using the Gaussian and mean curvatures, where one of ten surface types is assigned to each pixel. The connected regions of identical surface type pixels provide the first level grouping, a rough initial segmentation by surface types.

The output of these two independent channels are combined in the integration module. An edge type is assigned to each edge segment based on its location in the segmented image by surface types. Region merging, based on global surface fit error, is attempted for those regions inside a large region enclosed by edge segments. A region is split if the global surface fit error for the region is larger than an error budget. The output of the integration is a set of surface cluster graphs (SCG), where a node corresponds to a surface patch generated by the segmentation and an arc describes the relationship between two surface patches. A SCG roughly corresponds to a distinct object in the scene.

Finally, recognition is tried for each SCG. The surface characteristics of each surface in the SCG provides initial pruning in the hypotheses generation of object models. Each hypothesis is tested by computing the viewing transformation between the hypothesized model and the SCG. Recognition is successful when a hypothesis can be transformed into a correspondence with the image features.

CHAPTER 1
INTRODUCTION

Industrial applications of computer vision technology have proliferated in recent years [Tech Tran 85]. Computer vision has emerged as one of the newest technologies for use in the factory of the future, as well as in the factory of today, to meet the ever increasing demands of productivity, quality control, safety, and reliability in the manufacturing industry. This tremendous interest in computer vision from the manufacturing industry is a result of research and development efforts in academia and research institutes around the country over the past 20 years [Brad 82, Besl and Jain 85]. However, most of the current machine vision applications are restricted to the inspection, measurement, identification, and handling of simple parts [Tech Tran 85]. Much work remains to be done for a computer vision based system to determine general 3-D workpiece identification, position, orientation, and shape. The future robotic and automated flexible manufacturing systems will require a computer vision system which can understand its environment to perform various manufacturing tasks.

One reason for this slow progress of applications of full-fledged computer vision technology toward an automated

manufacturing system is that many manufacturing tasks require sophisticated and difficult visual interpretation, yet current computer vision technology can not fulfill those requirements, much less the requirements of low cost, high speed, accuracy, and flexibility.

Computer vision (hereafter synonymously called machine vision, robot vision, or image understanding in this dissertation) is an amalgam of diverse fields, among which is image processing, pattern recognition, and artificial intelligence. Vision involves the physical elements of illumination, geometry, reflectivity, and image formation, as well as the intelligence aspects of recognition and understanding. An image is generated by complicated interactions of these physical elements, that is to say, it is an encoding of the interactions among these physical elements. Much of current computer vision research addresses the decoding of these interactions, namely extraction of the physical properties of the scene (depth, surface orientation, reflectance, etc.) from images by using only a few general assumptions about the scene domain [Brad 82, Barr and Tene 83, Besl and Jain 85].

Recognition or understanding cannot be achieved by the direct use of an image which is nothing more than an array of numbers representing the brightness of a scene, though they are not random. It requires a high level symbolic description of the scene. The transformation from the input image to high

level symbolic description can not be achieved in one step. It is a complex task and it is usually accomplished through a hierarchy of representations and processes, where an image is initially transformed into edges and where the edges, in turn, are grouped into more meaningful structures, and so on. The abstraction level increases as the processing goes up the hierarchy. Figure 1.1 shows this abstraction hierarchy of representations from image to high level symbolic description.

This hierarchy of processes is sometimes loosely classified as low-level vision, mid-level vision, and high-level vision. The boundary between the levels has never been clearly drawn.

Low-level vision directly deals with image-level operations and is concerned with robust extraction of significant local features such as edges, highlights, or surface orientation using operations including smoothing, filtering, edge detection, and intrinsic image calculation and segmentation of the image into homogeneous regions. The role of the low-level vision process is to generate domain-independent rich descriptions of an input image to aid in the perceptual grouping process of mid-level vision.

The role of the mid-level vision process is to organize these rich descriptions of low-level vision output into perceptually more meaningful and compact structures. This domain independent structuring of important features produces a rough sketch (global cues) of a scene viewed or an accurate

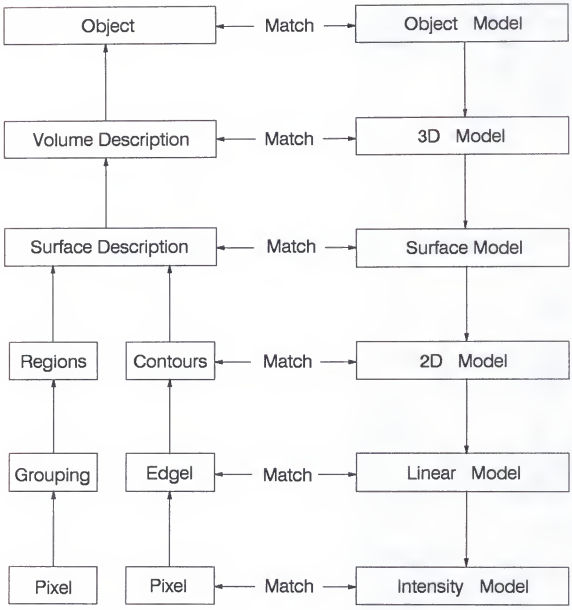


Figure 1.1 Abstraction hierarchy of image description

sketch of the scene if the image formation process were ideal, which will enable the high-level tasks to more easily determine the identification, localization, and orientation of objects.

High-level vision is concerned with the reasoning aspects of visual processing and includes tasks such as naming and categorizing the observed stimuli, model-based vision, and construction and manipulation of data bases of expected objects, based on immediate experiences and contextual cues.

Most of computer vision research has been devoted to either low-level vision processes or high-level symbolic reasoning processes, assuming that mid-level vision processes are taken care of by high-level or low-level vision processes. Recently, this mid-level vision process has received more attention [Lowe 87, Fisc and Boll 86]. Our research is in line with this emphasis on the mid-level vision.

This classification of vision processes has a very strong relationship on how to implement the control structure of a machine vision system for a very sophisticated, real world vision task. Low-level vision has been clearly defined as bottom-up or data-driven processes, and high-level vision as top-down or goal-driven. Most of the current systems invoke high-level vision too early, with imperfect structuring of image features making it useful only in very restricted domains.

1.1 Motivation of the Research

The ultimate goal of a computer vision system is to be able to partition the image into distinct regions, each of which has different physical properties, and to understand the relationships among them. This is a segmentation problem. There has been two approaches to the segmentation problem: an edge-based approach and a region-based approach. These dual approaches attack the same problem with different methods and philosophy.

The edge-based approach [Brad 82, Cann 86, Lowe 87] looks for places where significant intensity changes take place, with the hope that edges will enclose distinct regions. Unfortunately, edge detection invariably produces spurious edges as well as real edges and misses some real edges. Some researchers [Boll and Cain 82, Neva and Babu 80] attempt to bridge the missing edge gaps by ad hoc methods and others try to use incomplete data in recognition, which results in combinatorial explosion of the search space if the number of objects in the model data base becomes large. Some recent attempts [Low 87] to use a perceptual grouping process using proximity, collinearity, and parallelism seem promising. Still these decisions are based on local observation and not on global observation.

A region-based approach starts with some seed regions and grows the regions using homogeneity constraints [Hara and Shap 85, Besl and Jain 88]. The result depends on the seed regions

chosen and the region growing methods. It does produce disjoint divisions of the given image but often either oversegments or undersegments, hence some region boundaries may not correspond to the physical boundaries of the objects in the scene. The image we are using in this research is a surface orientation map generated by a photometric stereo method [Wood 80]. Unlike gray level images, the surface orientation map provides explicit three-dimensional information of the scene. It has the same iconic structure, except that each entry is a surface normal of the scene. We attempt to segment and recognize a scene of three-dimensional objects.

Our main research goal is to fuse the two completely independent outputs of an edge detection process and a surface segmentation process based on the signs of mean and Gaussian curvature values computed at each pixel into a reliable and coherent surface patch description that will facilitate object recognition. Figure 1.2 is the proposed object recognition system architecture following the abstraction hierarchy of Figure 1.1. Edges are detected by a traditional method. However, unlike previous approaches, which immediately try to attach semantic meanings, such as collinearity, parallelism, convexity, and concavity to the detected edges, the meaning of an edge decision is based on the segmentation result. Hence, it is much more reliable. Likewise, the meaning of a region is strengthened by the adjacent edges surrounding the region.

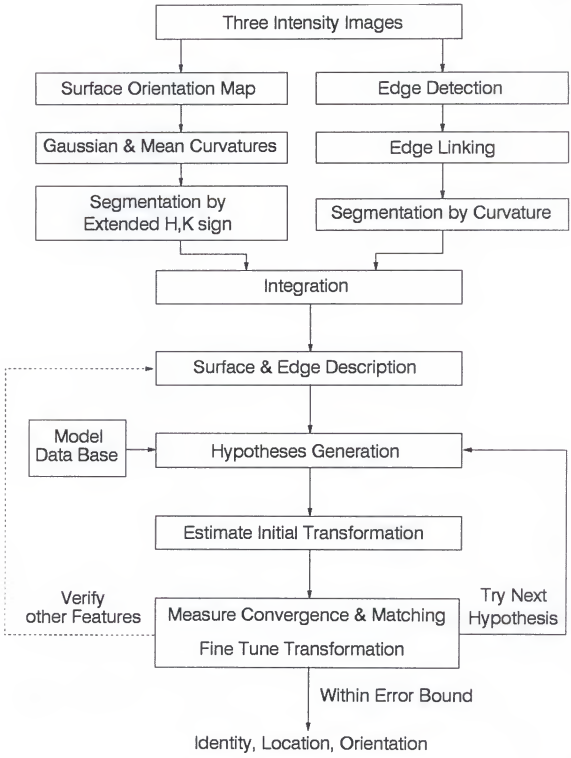


Figure 1.2 System architecture for 3-D object recognition

Our research emphasizes mid-level vision, whose role is to provide high-level vision with as much information as possible about the scene contents in a structured way so that high-level vision can use this information to prune the search space of the model data base. An aggregate of highly structured (meaningful structure) information, such as surface patches, can directly index into the model data base of a given application domain. The integration of surface based segmentation and edge detection output provides powerful clues to high-level task of object recognition.

1.2 Overview

The dissertation is organized into six chapters. In chapter 2, several edge detection methods are first discussed. A Canny edge detector [Cann 86] is used to detect edges. Extracted edges are then connected into edge contours. Important image features, such as corners and inflection points, are identified, based on the curvature values along the edge contours. These feature points are used to segment edge contours into straight line segments and conic section segments. The segmented edge contours provide one source of image segmentation.

Chapter 3 starts with a discussion of the photometric stereo method needed to derive a surface orientation image. Important differential geometry concepts of principal curvatures, mean and Gaussian curvatures, and the

classification of the surface types using these curvature values are then introduced. The computation of these curvatures from the surface orientation map is then discussed. The labeled surface type image is generated using the signs of the Gaussian and mean curvatures, where one of the ten surface types is assigned to each pixel. The connected regions of identical surface type pixels provide the first level grouping, a rough initial segmentation.

Chapter 4 discusses the integration of information from the two separate channels of the segmentation process. An edge type is assigned to each edge contour segment, based on its location in the segmented image by surface type. Region merging, based on global surface fit error, is attempted for those regions inside a large region enclosed by edge contour segments. A region is split if the global surface fit error for the region is larger than an error budget. The output of the integration is a set of surface cluster graphs (SCG) where a node corresponds to a surface patch generated by the segmentation and an arc describes the relationship between two surface patches. An SCG roughly corresponds to a distinct object in the scene.

In chapter 5, object recognition, using the surface cluster graphs, is discussed. A surface cluster graph is used to generate the initial hypotheses of object models. The initial pruning of the hypotheses is accomplished using a surface classification network. The adjacency and geometric

constraints of the surface cluster graph provide additional sources for further pruning. The hypotheses are then tested for acceptance or rejection by computing the viewing transformation between an object model and a surface cluster graph and projecting the model into the image to match the observed image features.

Finally, chapter 6 summarizes the contributions of this research and indicates future directions of research.

CHAPTER 2
EDGE-BASED SEGMENTATION

The goal of image segmentation is to partition an image into disjoint regions which may correspond to objects, surfaces, or parts of objects [Hara and Shap 85]. These regions are internally homogeneous, but heterogeneous between adjacent regions, in some properties such as intensity, texture, surface characteristics and so on. One approach to image segmentation is to detect the boundaries which separate different regions. Boundaries of regions are usually manifested in an image as places where significant intensity changes occur. The place where this significant intensity change takes place is traditionally called an edge in image understanding. Hence edge detection is a central problem in image understanding.

The human vision system uses diverse visual cues such as stereopsis, texture, shading, color, contour, etc. for recognition and understanding of its environment [Brad 82, Barr and Tene 83]. The boundaries of objects are the single most prominent and important features used for the recognition and description of objects [Marr 80, Brad 82]. This observation is derived from many experiments done with the human vision system, which can recognize objects from only a

crude outline and communicate three dimensionality using simple line drawings.

From a practical point of view, the edge detection process serves to simplify the analysis of images by dramatically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries. Image data itself is too dense to be directly used for high-level tasks.

For these reasons, vision researchers have tried to develop an ideal perfect edge detector. Edge detection is the first stage toward the symbolic transformation of an image, and the remaining stages of the visual recognition process all depend on the results of the edge detection. For example, stereo analysis [Marr and Pogg 79] should first solve a correspondence problem between two images, which is extremely difficult to do. Edge information, especially that of corners and curvature discontinuity points, provides a good starting point for solving the correspondence problem. Similar arguments hold for motion analysis [Ball and Brow 82], where the features in a sequence of images should first be registered together for motion parameter computation.

A critical, intermediate goal of edge detection is the detection and characterization of significant intensity changes in an image. The ultimate goal of edge detection is the characterization of intensity changes in the image in terms of the physical processes that originated them. This is

yet an unsolved problem because various physical changes in a scene, such as depth discontinuity, surface orientation discontinuity, reflectance discontinuity, and illumination discontinuity, result in the same intensity change in the image.

2.1 Review of Edge Detection

Since edges are the places where significant intensity changes take place, spatial differentiation is required. Various edge detectors [Marr and Hild 80, Hara 84, Cann 86] are discrete approximations of the spatial differentiation.

Edges in real images are not ideal step edges for which the edge operators were designed. Rather, real edges look like a gradual ramp shape contaminated by noise. There have been two broad approaches to edge detection: first-derivative type operators and second-derivative type operators. Edges in real images can be detected either as maxima of a first-order derivative or as zeros of a second-order derivative. Since edges can occur in any two dimensional direction, these operators can further be divided into directional or nondirectional types.

The response of first-derivative type edge operators to these real edges covers a relatively broad region flanking the edge. One of the great disadvantages of first-derivative type operators is that another process of thinning or peak detection of this wide edge response is required, which

degrades localization in addition to the additional computation time. Second-derivative operators, on the other hand, respond with a zero crossing at edge location. Therefore, edges are much easier to localize but it is more difficult to detect intensity changes because double differentiation is doubly sensitive to noise.

The first gradient operator called Roberts's cross operator was used by Roberts [Robe 63]. This operator involves only 4 pixels and is therefore extremely sensitive to noise and surface irregularities. With the noise sensitivity in mind, the extension of Robert's operator was proposed by Sobel and Prewitt [Ball and Brow 82, Prew 70]. These operators perform local averaging around the center point, thus they tend to reduce the effects of noise.

Marr and Hildreth proposed using the Laplacian of Gaussian (LOG) as an edge detector [Marr and Hild 80], which can be approximated by a Difference of Gaussian (DOG) edge detector, believed to be in the human vision system.

LOG has some drawbacks as well as some merits. The advantages of LOG include: 1) Computations are efficient and simple. 2) Noise removal can be accomplished by the selection of standard deviation (σ) of Gaussian, and scale of edges can also be controlled by the same standard deviation. 3) Subpixel accuracy can be achieved by simple linear interpolation across zero crossing.

The drawbacks of the LOG operator are as follows: 1) LOG has a worse signal to noise ratio (SNR) and localization than the first derivative of Gaussian operator. 2) Zero crossings do not necessarily correspond to real edges in noisy real images. Since zero crossings do not contain edge contrast information nor edge direction information, the additional step of gradient computation is needed to compute the zero crossing slope and direction. 3) At vertices where an odd number of edges converge, the end of interior edges turns away from the corner. This is because the assumption of linear variation does not hold around corners.

Several researchers proposed surface fitting techniques for edge detection [Prew 70, Hara 84]. Here, an image is viewed as samples of underlying intensity surface. At each pixel, surface fitting is performed. Edge properties such as edge magnitude, position, and orientation are estimated from the modeled image surface. Since the basis functions used to model the surface are usually not complete, the properties apply only to a projection of the actual image surface on to the subspace spanned by the basis functions. Haralick [Hara and Wats 83] later extended this approach to classify intensity surface type, called "topographic primal sketch," into peak, pit, flat, valley, ravine, etc., which can be used to segment an image into regions of similar intensity characteristics.

Canny [Cann 86] formulated the edge detection problem as an optimization problem using the following three performance criteria.

(1) Good detection.

This criterion requires a low probability of missing a real edge and low probability of a false alarm, which corresponds to maximizing the signal to noise ratio.

(2) Good localization.

The points marked as edges by the operator should be as close as possible to the center of the true edge.

(3) A single response to a single edge.

When two nearby operators respond to the same edge, one of them must be considered a false edge.

The criteria (1) and (2) are conflicting because good detection requires a large operator size (hence high signal to noise ratio) but results in poor localization, and good localization requires a small operator, which degrades reliable detection. This is an uncertainty principle and must be compromised.

Canny then solves the optimization problem by numerical methods. The optimal edge detector thus derived has a slightly better performance than the first derivative of the Gaussian operator, which can be efficiently implemented because of the separability of the Gaussian kernel. There can

be many variations in the implementation of the Canny edge detector.

The directional second derivative of the Gaussian has better performance than the Laplacian. The two-dimensional Laplacian can be decomposed into two arbitrary directions. If we take one of the derivatives in the direction of the principal gradient, the operator output will contain one contribution that is essentially the same as that of the operator described above, and also a contribution that is aligned along the edge direction. This second component contributes nothing to localization or detection, but increases the output noise.

Two main observations argue in favor of a gradient type operator. (1) A gradient operator is more robust against noise because it uses only first-order derivatives and not second-order derivatives as in a zero crossing scheme. (2) A zero of a second-order derivative does not necessarily coincide with an extremum of the first-order derivative; we expect a lower proportion of false edges in a gradient scheme.

Edge detection is a very complex problem. Vision researchers now seem to agree, after long years of frustration in the pursuit of an ideal edge detector and better understanding of image formation process, that edge detection is an ill-posed problem [Bert 88]. This is because significant intensity changes (corresponding to real world edges) occur in so wide a range of scales (depending on illumination,

viewpoint, surface orientation, material of object, etc.) that no single edge detector can detect all these changes simultaneously. For a competent vision system, we need other sources of information such as range, surface orientation, or high level knowledge to interpret edges.

2.2 Edge Linking

The first level symbolic transformation of an image is accomplished by the convolution of the image with one of the edge detectors of the above section. The transformed images (edge magnitude, edge direction) still have an iconic structure in the sense that instead of brightness in the scene, each pixel now represents a degree of edginess by its magnitude or an edge direction. Edge information, by itself, is of little use in a vision system. The next stage of processing imposes structure on the individual edge points by grouping them to form extended edges. Previous approaches to this problem broadly fall into two categories--global and local. Global strategies have been formulated as a graph search for minimum cost paths where the cost of connecting two edge points may depend on their proximity, relative orientation, and contrasts, among other factors. Dynamic programming and heuristic search are frequently used tools for this purpose. Local approaches, on the other hand, use local connectivity and directionality when edge points are linked.

To deal with ambiguity and noise in a vision system, a class of mechanisms called relaxation labeling was proposed. A probabilistic relaxation method [Zuck et al. 77] was proposed for edge enhancement and linking. Each pixel has m+1 labels. Each of m labels corresponds to the direction of the edge at the pixel with one additional label to indicate no edge presence. For each pixel i, the probability of label λ is defined. The initial value of this probability $P_i(\lambda)$ is calculated by applying an edge operator to the pixel. Each probability is updated according to its neighbor's probability. For each pair of neighboring pixels i,j and each pair of labels λ, λ', the compatibility function $r_{ij}(\lambda,\lambda')$ is defined, denoting a measure of compatibility between λ at i and λ' at j. The probability $P_i(\lambda)$ is strengthened by λ' of a neighboring pixel of similar direction and weakened by perpendicular direction.

The Hough transform [Duda and Hart 73] is another popular method for extracting global information in incomplete and noisy data such as the edge detection output. Most of the edge detector outputs suffer from the problems of fragmented edges, false edges, and missing edges due to the embedded noise in the image. The Hough Transform reparameterizes image space into parameter space so that edge points which lie along the same line will have the same coordinates in parameter space. By transforming all edge points in this way and looking for clusters which lie at the same locations in transformed space,

it is possible to search efficiently for all sets of collinear points. Parameter space transformation provides robustness against missing edges and spurious edges. But the Hough transform has several inherent drawbacks. First, the result is sensitive to the quantization of parameter space and has problems with clustering entries corresponding to nearly collinear points. Second, the transform finds collinear points without regard to contiguity. Thus the position of a best fit line can be distorted by the presence of unrelated edge points in other parts of the image. A related problem is that of meaningless groups of collinear points detected as a line. Although the Hough transform can be useful in detecting the presence or absence of strong features in a given image, its direct usage is limited by the fact that it does not provide any information as to the location of a line segment being considered.

Edge linking methods can improve the edge detection output, but these methods can not completely solve the inherent problems of missing edges and the spurious edge problem of edge detection. More global information is needed to resolve these issues. Figure 2.1 shows the block diagram of the edge-based segmentation used in this dissertation. An image is convolved with the first derivative of Gaussian to detect edges. Peak detection is performed on the edge detection output to produce edges of single pixel width. The edge points are then linked to generate edge contours. Edge

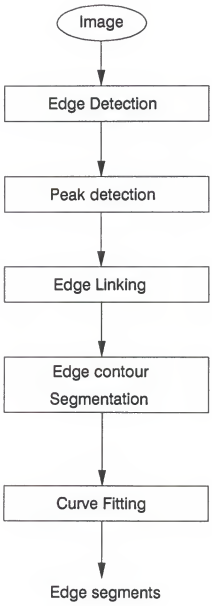


Figure 2.1 Edge-based segmentation

contours are segmented at corners and inflection points and fitted to lines or conics to generate edge segments. Edge segments provide initial region boundaries. Edge segments rarely enclose regions and there are inevitable gaps between edge segments. During integration of edge-based and surface-based segmentation, we can use the mathematical description of the edge segments to fill the gaps between edge segments and extend the edge segments to enclose regions. In the following sections we will discuss edge-based segmentation in detail.

2.3 Implementation of the Edge Detector

Since edge detection is a differentiation process, smoothing is essential for robust and reliable edge detection in noisy image data. There are many smoothing filters, but a Gaussian filter is found to be the only filter exhibiting low bandwidth for a given filter size, which is best for smoothing [Marr and Hild 80]. An image is first convolved with a Gaussian smoothing filter, then a gradient operator is applied at each pixel. By the associativity property of the convolution theorem, these two operations can be combined into a single operation by convolving the image I with the first derivative of Gaussian.

$$\begin{aligned} \nabla(G * I) &= (\nabla G) * I \\ &= \left(\frac{\partial G}{\partial x} * I\right)\mathbf{i} + \left(\frac{\partial G}{\partial y} * I\right)\mathbf{j} \\ &= \nabla_x \mathbf{i} + \nabla_y \mathbf{j} \end{aligned} \qquad (2.1)$$

where G is a Gaussian with a standard deviation σ and **i** and **j** are unit vectors in x and y coordinate directions.

$$G(x,y) = K \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{2.2}$$

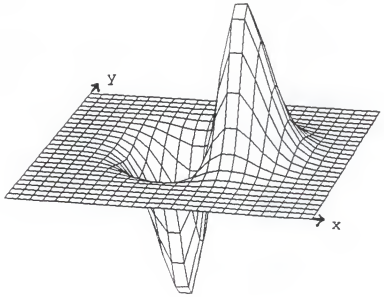
The convolution kernel of the first derivative of the two-dimensional Gaussian can be decomposed into a product of one-dimensional Gaussian and its derivative. Fast convolution is achieved by two one-dimensional convolutions instead of a two-dimensional convolution.

$$\begin{aligned} \frac{\partial G(x,y)}{\partial x} &= G_x(x,y) = -\frac{Kx}{\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \\ &= \left(-\frac{Kx}{\sigma^2} e^{-x^2/2\sigma^2}\right)\left(e^{-y^2/2\sigma^2}\right) \end{aligned} \tag{2.3}$$

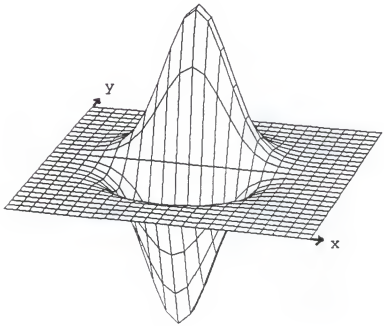
$$\begin{aligned} \frac{\partial G(x,y)}{\partial y} &= G_y(x,y) = -\frac{Ky}{\sigma^2} e^{-(x^2+yxp2)/2\sigma^2} \\ &= \left(e^{-x^2/2\sigma^2}\right)\left(-\frac{Ky}{\sigma^2} e^{-y^2/2\sigma^2}\right) \end{aligned} \tag{2.4}$$

The scale factor K is adjusted, depending on the precision available in a particular computer. Fig. 2.2 shows the three-dimensional graph of this operator.

Since the gradient type edge detector also responds to the gradual shading change, the image is first truncated at several low order bits before gradient computation to suppress



(a)



(b)

Figure 2.2 First derivative of Gaussian edge detector
(a) row operator (b) column operator

its response to gradual shading change. Discrete approximation of the above operator will be samples of the first derivative of Gaussian in equation (2.3). In order to achieve accuracy, we use the average value of the first derivative of the Gaussian at each pixel size interval, which is equivalent to its integrated value over a unit pixel.

As the decomposition of the kernels indicates, ∇_x computation is carried out by convolving the image with the one-dimensional first derivative of Gaussian in column operation followed by Gaussian smoothing in row operation. Similarly ∇_y is computed in row-wise first derivative convolution, followed by column-wise Gaussian smoothing.

The gradient of image I is a vector whose magnitude at a given point is the maximum rate of change of the intensity surface I at that point and whose direction is normal to the underlying intensity change direction (edge direction). The edge magnitude ∇_{mag} (gradient magnitude) and edge direction α (gradient direction) are computed as

$$\nabla_{mag} = \sqrt{\nabla_x^2 + \nabla_y^2} \approx |\nabla_x| + |\nabla_y| \qquad (2.5)$$
$$\alpha \text{ (angle)} = \arctan(\nabla_y / \nabla_x)$$

As we noted in section 2.1, the convolution output of the gradient operator has broad response across the edge, which must be thinned or peak detected by eliminating the edge points which are not local maxima. Thinning is not used,

because it degrades the accuracy of localization. Peak detection is performed by eliminating the pixels if they are not local maxima. For every nonzero edge pixel centered at a 3 × 3 window, its neighborhood is examined to decide if the center pixel is a local maximum or not. The two adjacent edge magnitude values along the gradient direction are computed by interpolation using the gradient directions and edge magnitudes of the pixels in the window. If the two adjacent gradient magnitudes are less than the magnitude of the center pixel, the center pixel is retained as a local maximum. Otherwise the center pixel is eliminated. After the peak detection, edges become single pixel width.

Edge contour tracing

The peak detected convolution output is used to generate edge contours. An edge contour is an ordered list of connected edge points. Since edge detection also responds to any intensity change in image, the use of thresholding is inevitable to eliminate edge points whose magnitudes are small. Selecting an appropriate threshold value is very difficult. A high threshold will eliminate noisy edges but it will also remove some of the real edges. On the other hand a low threshold will retain most of the real edges but it also invites noisy edges. Simple thresholding on the edge detection output results in a streaking problem of edge contours, breaking up of an edge contour caused by the edge

detector output fluctuating above and below the threshold due to noise.

Patching up the broken contour by a later process is very difficult. In order to avoid this streaking problem, the whole edge contour is traced first, then thresholding is performed on the whole contour if the average edge magnitude of the whole contour is below a threshold.

Finding edge contours in the peak detected edge image becomes easy because only local maximum edge pixels are left. First, an island of connected pixels whose magnitude are above a threshold is used to start tracing an edge contour. As each edge point is traced, it is marked by the current edge contour number to avoid tracing it again. At a junction point the tracing continues in the direction which is similar to that of the previous pixel direction and the junction point is put on a stack to resume tracing to other branches at the junction. The tracing tries to fill one pixel gap by extending the tracing one pixel. The tracing stops either when no more edge points are nearby or when previously traced edge points are encountered.

A traced edge contour is discarded if the average edge magnitude is below a given threshold. But if the length of the contour is long enough (30 pixels), it is retained in the background data base for use in the integration module. Edge contour information is stored in the edge contour frame of

Figure 2.3. An edge contour frame contains ten pieces of information about the contour:

(1) length: the number of pixels in the contour

(2) average_magnitude: the average edge magnitude of the contour

(3) start_coord: starting coordinates of the contour

(4) end_coord: ending coordinates of the contour

(5) open_status: open or closed status of the contour. If the tracing returns to the starting pixel, the open_status slot is filled with the closed value. Otherwise it is filled with the open value.

(6) adjacent_contour: a list of adjacent edge contours. If the tracing stops at another traced edge contour, the adjacent_contour slot is updated with the edge contour that was encountered. The list of adjacent contours provides useful information to find the junctions where two or more lines meet. Junctions and junction types will be discussed in chapter 4.

(7) chain_list: an ordered list of 8-neighbor chain codes. The coordinates data are recovered from the chain list to fit lines or conic sections.

(8) orientation_list: an ordered list of edge direction. This data is used to compute curvatures along the contour to segment the contour.

(9) knot_points: points on the contour where tangent discontinuity (corner) or curvature discontinuity

(inflection point) occur. An edge contour is divided at knot_points. Edge segments between knot points are represented as lines or conic sections. This slot is filled after edge contour segmentation.

(10) edge_segments: An edge contour is divided into edge segments where each segment is represented by a line, an ellipse, or a circle parameters after edge contour segmentation. This slot contains a list of pointers to the edge segments. This slot is also filled after edge contour segmentation.

```
(CONT_11  (length  56)
          (average_magnitude  227)
          (start_coord    (x1  y1))
          (end_coord      (x2  y2))
          (open_status    open )
          (adjacent_contour  (CONT_26 CONT_30 ...))
          (chain_list     chain_11 )
          (orientation-list  orient_11 )
          (knot_points    (k1 k2 ... ))
          (edge_segments (e1 e2 ... )))
```

Figure 2.3 An example of an edge contour Frame

The edge contour frame provides all the necessary information for edge contour segmentation. Edge contours are

input to the module of analytic curve descriptor where each edge contour is segmented and fitted according to curvature values.

2.4 Description of Edge Contours by Straight Lines and Conic Sections

2.4.1 Segmentation of Edge Contours

The next processing stage is to represent each edge contour at more qualitative and quantitative abstraction levels. For example, on a qualitative level we would like to know which portions of an edge contour are linear or curved and on a quantitative level we would like to know the line equations for linear parts and the degree of curvature for curved parts. We would like to derive appropriate mathematical expressions describing each segment of an edge contour. Such expressions are far more compact than discrete forms, and amenable for matching two such representations. We also want to discover instances of certain features (e.g., straight lines, corners, inflection points) which play an important role in image segmentation. The segmentation of edge contours serves another purpose in our research. An edge contour may be long separating many different regions in the image. It is difficult to represent and use the whole contour during region analysis, which will be discussed in Chapter 4. Segments of edge contours can serve as the initial region boundaries.

Now we have to consider two issues encountered when attempting to fit curves to data.

1. We must choose a family of curves to be fitted.

2. We must decide on how to choose the knot points, given our choice of curves.

We decided to use conics for a variety of reasons. Conics are commonly used to represent general curves approximately. The coefficients or parameters of conic sections are terse representations. Conics are often good models for physical curves such as the edges of manufactured objects.

A curvature-based segmentation of a curve has attributes which make it insensitive to changes in viewpoint. In the plane, the curvature is invariant with respect to rotation and translation, and curvature ratios are invariant with respect to scale. The use of tangent discontinuity (usually called a corner) and the zero crossing of curvature (inflection point) as segmentation points provides insensitivity with respect to the projection of a plane curve oriented arbitrarily in space. Mathematically, tangent discontinuity is undefined in curvature, but in real images it is identified as a curvature extremum. A curvature extremum is a point on the contour where the curvature is locally maximum or minimum. A curvature extremum is not necessarily a tangent discontinuity (see the extremum point a in Figure 2.8 on page 52 which corresponds to segmentation point 3 in Figure 2.7 (b)). An

extremum caused by tangent discontinuity should have zero curvature (straight line) segments on both of its sides (see the extremum point b in Figure 2.9 on page 53 which corresponds to the segmentation point 7 in Figure 2.7 (b)). We will use tangent discontinuity and curvature zero crossing as the contour segmentation points. Some authors [Fisc and Boll 86] use curvature extrema in their segmentation. But curvature extrema are not always invariant to rotation. In general, the human visual system is very sensitive to the curvature of lines, being able to reliably discriminate spatial differences of a few arc seconds.

One of the most frequently used method to segment an edge contour with line segments is a successive subdivision method. It breaks a given contour at the point most distant from the straight line which passes the two endpoints of the given contour. Each newly obtained subcontour is tested to see if it can be fitted by a straight line within a prescribed error limit or if the length of the subcontour has fallen below a prescribed minimum length. The subcontour is accepted as one line segment when one of the above criteria is satisfied. If a subcontour cannot be fitted with a straight line, the subcontour is successively subdivided at another breakpoint. Although this method is powerful for segmenting a contour consisting of only straight line segments, it can not detect curvature zero crossings.

2.4.2 Curvature Based Segmentation of Edge Contours

We decided to use curvature zero crossing and tangent discontinuity as our segmentation points to segment an edge contour. In this section we will discuss how to find those points. An edge contour frame generated during edge contour tracing contains a list of octal chain codes and the corresponding edge orientation list. The list of edge orientations is in tangent versus arc length format called ψ-s representation [Ball and Brow 82]. The solid curves in Figures 2.8, 2.9, and 2.16 are ψ-s representations of the cylinder boundary starting from the segmentation point 1 in Figure 2.7 (b), of the stair boundary starting from the segmentation point 5 in Figure 2.7 (b), and of the bulb boundary starting from the point 1 in Figure 2.15 (b).

A ψ-s representation has some interesting properties. Translation of a contour in the image plane does not affect its ψ-s plot. Its rotation corresponds to a shift along the ψ-axis by the angle of rotation, and the change of its scale corresponds to a proportional stretching or shrinking of the s-axis. The slope of the ψ-s curve, $d\psi/ds$, gives the curvature of the corresponding point on the edge contour. Furthermore, a straight line becomes a horizontal line (solid segment C in Figure 2.9 corresponds to edge segment C of Figure 2.7 (b)) and a circle becomes a straight line with a nonzero slope (solid segment C in Figure 2.16 corresponds to edge segment C of Figure 2.15 (b)), where slope is

proportional to the curvature of the circle. This representation does not solve the contour segmentation problem, but it makes segmentation easy to solve by reducing a circle or circular arc detection to a straight line detection in ψ-s space.

There is one minor problem to using edge orientation data directly. The orientation data are samples of tangents along the contour, but the sampling distance is not uniform. As a result, the curvature computation is inaccurate. This is due to a square grid of the image plane. For example, horizontal or vertical travel is 1 pixel distance, but diagonal travel is $\sqrt{2}$ pixel distance. Furthermore, tangent data is relatively noisy. Uniform sampling is carried out after orientation data is smoothed by a Gaussian filter.

The uniform sampled ψ-s data is then convolved with the first derivative of a Gaussian to estimate the curvatures along the contour. Dashed curves in Figures 2.8, 2.9, and 2.16 are curvature graph of the cylinder boundary, the stair boundary, and the bulb boundary respectively. Ideally, if we plot curvature as a function of arc length, a horizontal line of zero offset indicates the presence of a straight line (see the real curvature corresponding to interval 6 of Figure 2.9) and a horizontal line of nonzero offset indicates the presence of a circle whose radius is inversely proportional to the curvature value of the line (see the real curvature corresponding to interval 5 of Figure 2.16). A zero crossing

with two nonzero horizontal lines of opposite sign means an inflection point (see point b in Figure 2.16). We are mainly interested in locating tangent discontinuities and inflection points where curvature zero is located.

First, we identify three interval types in curvature output: zero curvature, positive curvature, and negative curvature intervals. Since edge direction is invariably noisy, an absolute curvature value below a prespecified threshold (0.015) is considered to have a zero curvature. The curvature output is examined point by point starting from the first point. The curvature value of the first point decides the first interval type. The interval type remains the same until a point with different curvature value is encountered. The interval type and length are then recorded. For each nonzero curvature interval, an absolute maximum curvature and the average curvature of the interval are also recorded. A new interval type then starts and the same process is repeated until all the points in the edge contour have been examined. Shaded horizontal bars in Figures 2.8, 2.9, and 2.16 show the detected interval types of the curvatures of the cylinder boundary, the stair boundary, and the bulb boundary.

If a short zero curvature interval (less than 10 pixels) is between two nonzero curvature intervals of the same sign and similar average curvature, three intervals are merged into one interval (see interval 6 in Figure 2.16). This step removes the effect of noise or digitization.

Nonzero curvature intervals whose absolute maximum curvatures in the intervals are above a certain threshold (0.07) are identified first. These intervals may correspond to either corners or curved segments. If an interval is less than minimum length (30 pixels) it is considered as a corner candidate, otherwise it is considered as a curved segment. Corners are located if a zero curvature interval is between two corner candidate intervals (see intervals 5, 6, and 7 in Figure 2.9). A least squares line fitting is tried on the segment between two corner candidates.

An inflection point is identified if a short zero curvature interval (less than 10 pixels) is between two nonzero curvature intervals of different sign. A zero crossing between interval 1 and 3 in Figure 2.16 is an example of an inflection point which corresponds to segmentation point 2 in Figure 2.15 (b). Another curvature discontinuity point occurs when a long zero curvature interval (greater than 40 pixels) is bordering a long nonzero curvature intervals (greater than 40 pixels). The zero crossing between interval 1 and 2 in Figure 2.8 is this type of curvature discontinuity which corresponds to segmentation point 2 in Figure 2.7 (b). In either case, a segmentation point is identified at the zero crossing. A least squares conic fitting, or a line fitting is applied to the appropriate intervals to verify the interval type and to derive the mathematical descriptions of the segments. As we discussed in the introduction, edge contours

rarely encloses regions and there are inevitable gaps between edge contours. During integration we can use these mathematical description to fill the gaps between edge segments and extend the edge segments to enclose regions.

The remaining intervals whose length are greater than 40 pixels are fitted according to the interval types. In the current implementation, the rest of the intervals are fitted to lines with the successive subdivision method. Merging of two adjacent lines is tried if the two line parameters are close enough (orientation difference is less than 5 degrees) and fitting error is within an error tolerance. The fitting error tolerance is given as 0.5 pixel in current implementation.

If the fitting error for an interval is greater than an error tolerance, an interval is divided into two subintervals at the breakpoint which is farthest from the line connecting two endpoints of the interval. This process is repeatedly applied until the fitting error is acceptable or the interval is less than a minimum length threshold. Parameters from the conic fitting are accepted only for a circle or an ellipse, because a hyperbola, or a parabola, is difficult to interpret. A hyperbola, or a parabola, is approximated by circular arc parameters.

Edge contours rarely enclose regions and there are inevitable gaps between edge contours. During integration, we need the mathematical descriptions of each edge segment to

fill the gaps of some edge segments and extend the open edge segments to enclose regions. For this purpose, the edge segment frame of Figure 2.4 is instantiated, and appropriate parameters are recorded for each fitted edge contour segment. An edge segment frame contains eight pieces of information:

(1) curve_type: This slot indicates the curve type of the edge segment.

(2) start_coord: This slot contains the starting coordinates of the edge segment.

(3) end_coord: This slot contains the ending coordinates of the edge segment.

(4) parameter: This slot contains line, ellipse, or circle equations depending on the curve type.

(5) adjacent_segments: This slot contains pointers to one or two neighbors of the edge segment.

(6) part_of: This slot indicates the original edge contour of which the edge segment is a part.

(7) edge_label: This slot indicates whether the edge segment is convex or concave. Edge type is assigned and used during the integration phase which will be discussed in Chapter 4.

(8) region_numbers: This slot contains the regions of which the edge segment is one of the boundaries of the regions. Region numbers are assigned and used during the integration phase which will be discussed in Chapter 4.

```
(Edge_segment21     (curve_type  (line, ellipse, circle arc))
                    (start_coord  ( x2 y2 ))
                    (end_coord    ( x3 y3 ))
                    (parameter    ( a1 a2 a3 a4 a5 ))
                    (adjacent_segments  (edge_segment22 ..))
                    (part_of      ( CONT_12 ))
                    (edge_label   ( convex concave ))
                    (region_numbers  ( reg_235 reg_210 )))
```

Figure 2.4 An example of edge segment frame

Figure 2.5 shows the flow diagram of edge contour segmentation discussed in this section.

2.4.3 Least Square Fitting of the Line

We need mathematical descriptions to connect missing edge segments and extend open edge segments to enclose the region boundaries. Line fitting is one of the most frequently used tools in image understanding. Standard line fitting minimizes the sum of vertical distances between the fitted line and the points. We will use the normal equation of a line where its best fit is defined by minimizing the sum of distances between the line and the points, not the sum of vertical distances. The following formulation allows line parameters to be easily updatable by adding or deleting extra points during fitting. This feature can improve the accuracy of line equations and

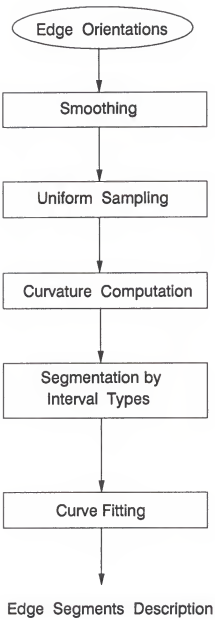


Figure 2.5 Flow diagram of edge contour segmentation

the locations of junctions which are intersections of lines. An edge detector response around a vertex is smoothly curving instead of showing sharp. By eliminating outlier edge points around a vertex during line fitting, accurate line parameters and vertex locations can be obtained.

Let the N given points be $((x_i,y_i),i=1,N)$ and let the straight line be represented by the following normal representation.

$$\rho = x\cos\theta + y\sin\theta \tag{2.6}$$

The perpendicular distance of a point (x_i,y_i) to a straight line is given by

$$d_i = \rho - x_i\cos\theta - y_i\sin\theta \tag{2.7}$$

Let D^2 be the sum of squared normal distance from N given points to the straight line, then

$$\begin{aligned} D^2 &= \sum_{i=1}^{N} (\rho - x_i\cos\theta - y_i\sin\theta)^2 \\ &= N\rho^2 + \cos^2\theta\sum_{i=1}^{N} x_i^2 + \sin^2\theta\sum_{i=1}^{N} y_i^2 \\ &\quad - 2\rho\cos\theta\sum_{i=1}^{N} x_i - 2\rho\sin\theta\sum_{i=1}^{N} y_i - 2\sin\theta\cos\theta\sum_{i=1}^{N} x_i y_i \end{aligned} \tag{2.8}$$

First the following terms are defined.

$$S_{xx} = \sum_{i=1}^{N} x_i^2, \quad S_{yy} = \sum_{i=1}^{N} y_i^2, \quad S_{xy} = \sum_{i=1}^{N} x_i y_i$$
$$S_x = \sum_{i=1}^{N} x_i, \quad S_y = \sum_{i=1}^{N} y_i \tag{2.9}$$

The parameters (θ, ρ) which minimize (2.8) are obtained from the following two equations

$$\frac{\partial D^2}{\partial \theta} = 2(\sin\theta\cos\theta[S_{yy} - S_{xx}] + \rho \sin\theta\, S_x - \rho\cos\theta\, S_y + [\cos^2\theta - \sin^2\theta]S_{xy}) = 0 \tag{2.10}$$

$$\frac{\partial D^2}{\partial \rho} = 2N\rho - 2\cos\theta\, S_x - 2\sin\theta\, S_y = 0 \tag{2.11}$$

Thus from equation (2.11) we have

$$\rho = \frac{S_x}{N}\cos\theta + \frac{S_y}{N}\sin\theta \tag{2.12}$$

Substituting ρ into equation (2.10) yields

$$(S_{yy} - S_{xx} + \frac{S_x^2 - S_y^2}{N})\sin 2\theta + 2(S_{xy} - \frac{S_x S_y}{N})\cos 2\theta = 0 \tag{2.13}$$

Solving equation (2.13) yields two solutions. The solution which minimizes equation (2.8) is the correct solution.

2.4.4 Least Square Fitting of the Conic Section

The general conic equation is given as

$$a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 = 0 \qquad (2.14)$$

Although there are six coefficients in eq. (2.14), one of them can be chosen arbitrarily, so that a conic has only five degrees of freedom. We assume, without loss of generality, that the sign of the first nonzero coefficient in eq. (2.14) is positive. The conic is an ellipse if the quantity $a_1a_3 - a_2^2$ is positive, a hyperbola if $a_1a_3 - a_2^2$ is negative, and a parabola if $a_1a_3 - a_2^2$ is zero. The conic is a circle if $a_1 = a_3$ and $a_2 = 0$. We obtain a five coefficient equation by normalizing the coefficients with the constant coefficient a_6.

$$a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + 1 = 0 \qquad (2.15)$$

Let $\underline{w}_i = [\ x_i^2 \quad x_iy_i \quad y_i^2 \quad x_i \quad y_i\ ]^T$ and

$\underline{a} = [\ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5\ ]^T$.

Then the optimum coefficient vector $\underline{a}$ based on n data points is that which minimizes (from (2.15))

$$\sum_{i=1}^{n} [\ 1 + \underline{a}^T\underline{w}_i\ ]^2 \qquad (2.16)$$

Upon defining A_n to be the matrix from n data points

$$\begin{bmatrix} w_1^T \\ w_2^T \\ \cdots \\ w_n^T \end{bmatrix}$$

b^n to be the n-component vector $[\ -1\ \ -1\ \ \ldots\ \ -1\ \ -1\ ]^T$, a least square solution to equation (2.16) is

$$\underline{a} = (\ A_n^T\ A_n\ )^{-1}\ A_n^T\ \underline{b}_n \qquad (2.17)$$

For the case of an ellipse, its center (x_c, y_c) is given [Ball and Brow 82]

$$x_c = (\ a_2a_5 - 2a_3a_4\ )/\ (\ a_2^2 - 4a_1a_3\ ) \qquad (2.18)$$
$$y_c = (\ 2a_5a_1 - a_2a_4\ )/\ (\ a_2^2 - 4a_1a_3\ )$$

The orientation of the ellipse is

$$\theta = 0.5\ \tan^{-1}\ (\ a_2/a_1 - a_3\ ) \qquad (2.19)$$

The major and minor axes of the ellipse are

$$\text{major axis} = \frac{-2G}{(a_1 + a_3) - \sqrt{a_2^2 + (a_1 - a_3)^2}}$$
$$\text{minor axis} = \frac{-2G}{(a_1 + a_3) + \sqrt{a_2^2 + (a_1 - a_3)^2}} \qquad (2.20)$$
$$\text{where}\quad G = 1 - (a_1x_c^2 + a_2x_cy_c + a_3y_c^2)$$

An edge segment of a nonzero curvature interval found in section 2.4.2 is fitted with the above procedure to find the parameters of the conic section. We use the parameters to extend the edge segment in order to enclose a region when one or both of the endpoints are not connected to the other edge segments.

2.4.5 Experimental Results

The edge detector and segmentation algorithms were implemented and tested for several scenes. Let's take an example to illustrate the edge-based segmentation discussed in this Chapter. Figure 2.6 (a) is the edge image of a stair and a cylinder. Figure 2.6 (b) is the thresholded edge image. Figure 2.7 (a) is the detected edge contours after the average gradient magnitude of the edge contour has been used for thresholding. Note that some edge gaps are filled and edges are extended by this thresholding scheme, providing more reliable edge contours. The solid curve in Figure 2.8 is the graph of the tangent of the outer contour of the cylinder versus arc length, starting from point 1 in Figure 2.7 (b). The dashed curve in Figure 2.8 is the corresponding curvature of the cylinder boundary. From the curvature output, eleven interval types are identified. Intervals are shown as a shaded horizontal bar. Negative curvature interval 1 corresponds to the bottom of the cylinder, which is fitted to an ellipse. Zero curvature interval 2 corresponds to the left side of the cylinder, which is fitted to a line. A tangent

discontinuity between intervals 1 and 2 is identified and marked at the segmentation point 2 in Figure 2.7 (b). Negative curvature interval 3 corresponds to the top of the cylinder, which is fitted to an ellipse. Another tangent discontinuity between intervals 2 and 3 is identified and marked at the segmentation point 3 in Figure 2.7 (b). The rest of the intervals do not qualify as a corner or a curvature discontinuity. They are fitted as line segments by the successive subdivision method. The solid curve in Figure 2.9 is the graph of the tangent of the outer contour of the stair versus arc length starting from point 5 in Figure 2.7 (b). The dashed curve in Figure 2.9 is the corresponding curvature of the stair boundary. Eight curvature extrema corresponding to eight corners are located between the zero curvature intervals. Each segment is fitted to a line segment within error tolerance. Figure 2.10 (a) shows the fitted straight lines and the conic sections for the edge contours segmented at corners and at curvature discontinuities. Note that each segment is now represented as appropriate mathematical equations. Figure 2.10 (b) is the image of the edge contours superimposed by fitted lines and conic sections. These edge contours are correctly segmented at appropriate segmentation points and we have an accurate mathematical description for each edge segment. Note many open edge segments.

Figures 2.11 (a) and 2.14 (a) are the edge images of a trough and stair and a bulb. Figures 2.11 (b) and 2.14 (b) are the thresholded edge images. Figures 2.12 (a) and 2.15(a) are the detected edge contours after the average gradient magnitude of the edge contour has been used for thresholding. Figures 2.12 (b) and 2.15 (b) are the identified segmentation points obtained through curvature analysis. Figure 2.16 is the graph of the tangent and curvature estimation of a bulb boundary. Five nonzero curvature intervals are identified. Intervals 5, 6, and 7 are fitted as an elliptic arc; the others are fitted as a parabola or a hyperbola. These segments are approximated by a circle fit. Detected lines and conic sections are shown in Figures 2.13 (a) and 2.17 (a). Superimposition of the fitted edge segments and original edge contours are shown in Figures 2.13 (b) and 2.17 (b).

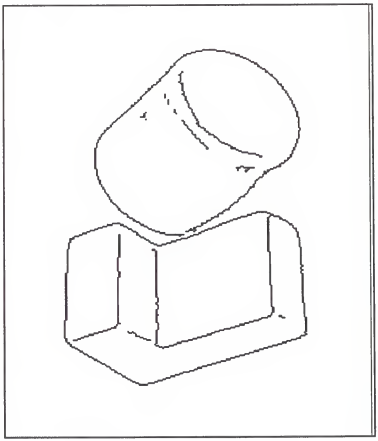
2.5 Summary

In this Chapter we discussed image segmentation based on edge detection. Edges are detected by convolving the image with the first derivative of Gaussian. After peak detection, edge points are linked to generate edge contours. Edge contours rarely enclose regions and there may be some gaps between edge contours. Since an edge contour may be the boundaries of many different regions in the image, it is difficult to represent and it must be segmented for subsequent region analysis. We presented edge contour segmentation based

on curvature analysis along the edge contours. Edge contours are segmented at corners and curvature discontinuities. A line or conic fitting is performed on the segmented contour to derive mathematical descriptions of each segment. Segmented edge contours serve as initial region boundaries.

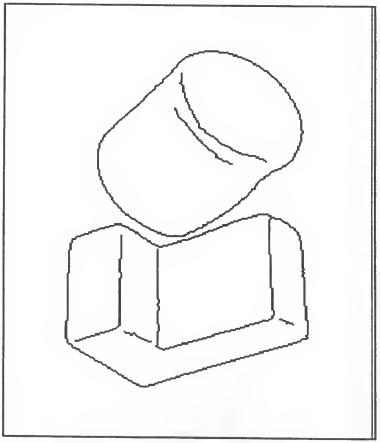
During the integration of edge-based segmentation and surface-based segmentation, we can use these mathematical description of each edge segment to connect broken edge segments, to find junctions, and to extend open edge segments for a reliable and coherent image segmentation.

(a) Edge magnitude image

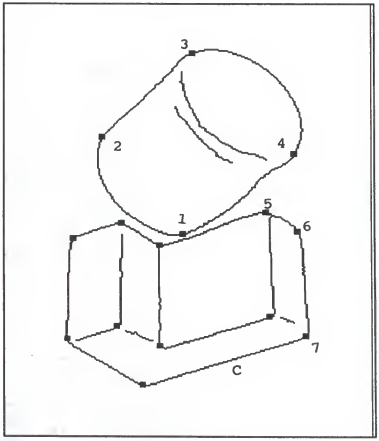


(b) Thresholded edge

Figure 2.6 Image of a stair and a cylinder



(a) Edge contour detected by contour thresholding



(b) Contour segmentation points

Figure 2.7 Detected segmentation points

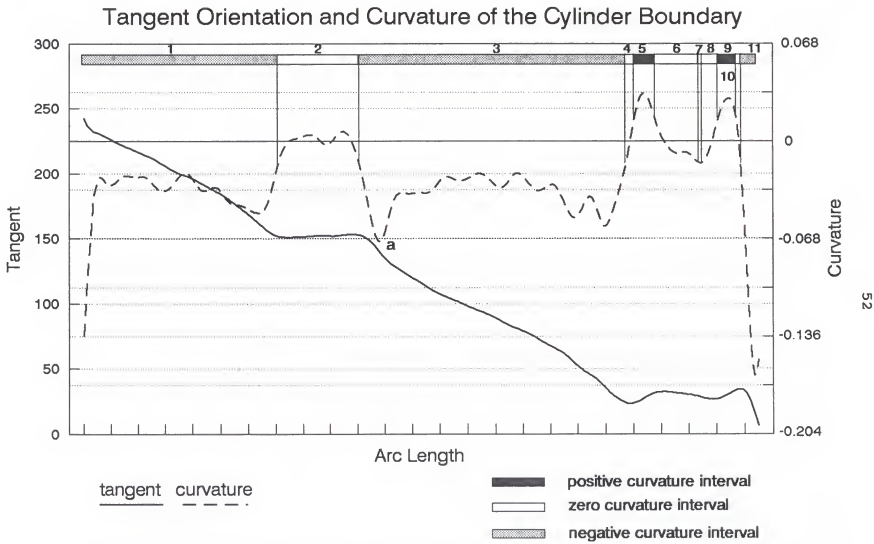


Figure 2.8 Tangent orientation and curvature of the cylinder boundary

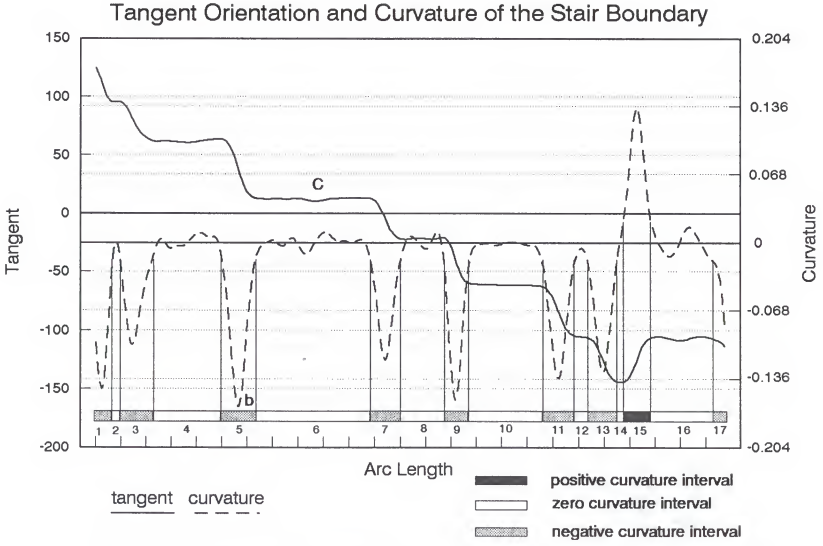
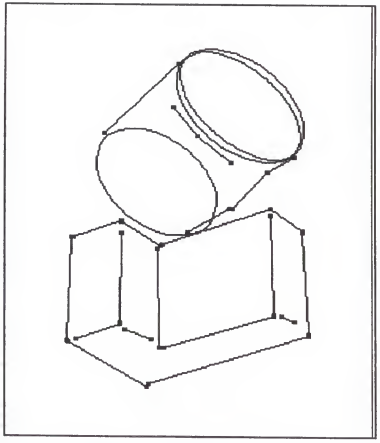
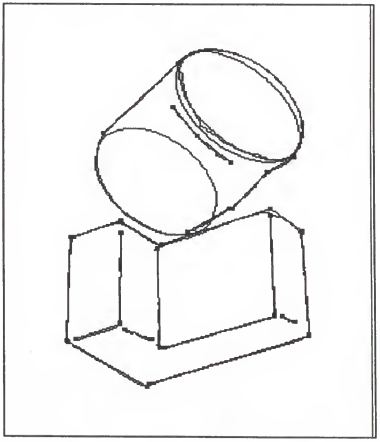


Figure 2.9 Tangent orientation and curvature of the stair boundary



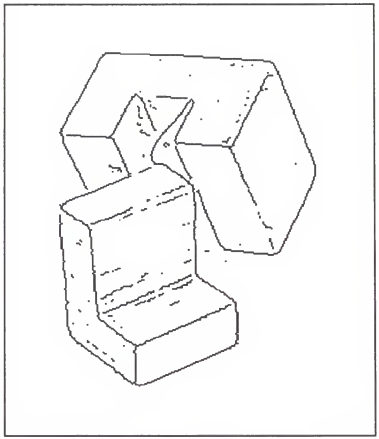
(a) Detected ellipses and straight lines



(b) Detected ellipses and lines superimposed on original image

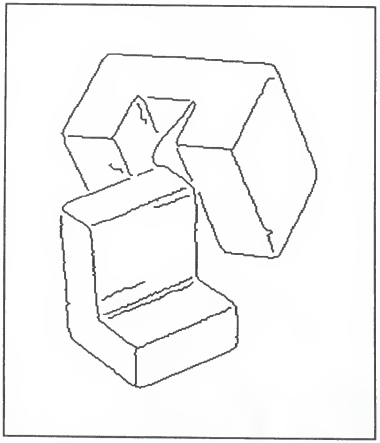
Figure 2.10 Detected lines and conic sections

(a) Edge magnitude image

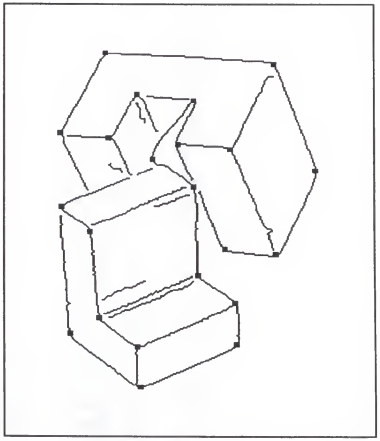


(b) Thresholded edge

Figure 2.11 Image of a trough and a stair

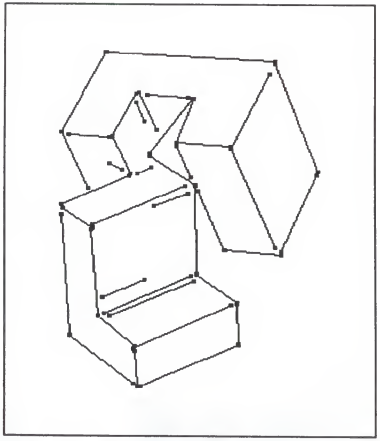


(a) Edge contour detected by contour thresholding

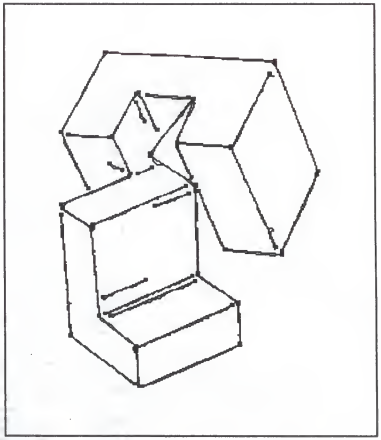


(b) Detected segmentation points

Figure 2.12 Detected segmentation points



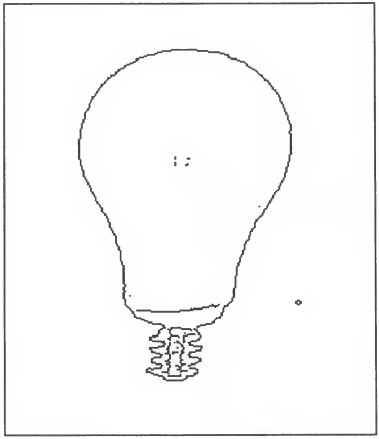
(a) Detected lines and conic sections



(b) Detected lines superimposed on the original image

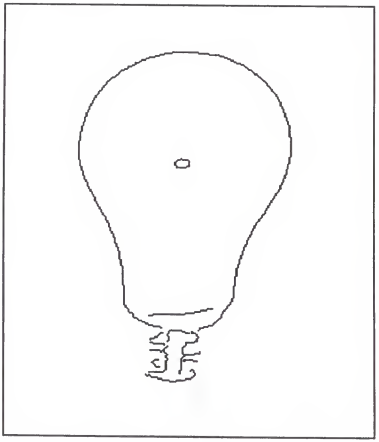
Figure 2.13 Detected lines and conic sections

(a) Edge magnitude image

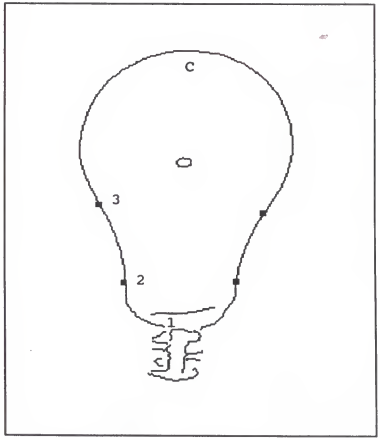


(b) Thresholded edge

Figure 2.14 Image of a bulb



(a) Edge contour detected by contour thresholding



(b) Detected segmentation points

Figure 2.15 Detected segmentation points

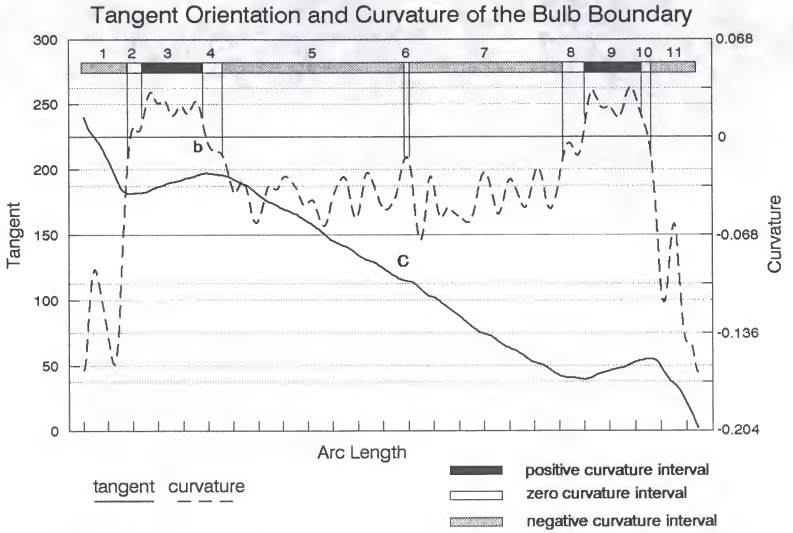
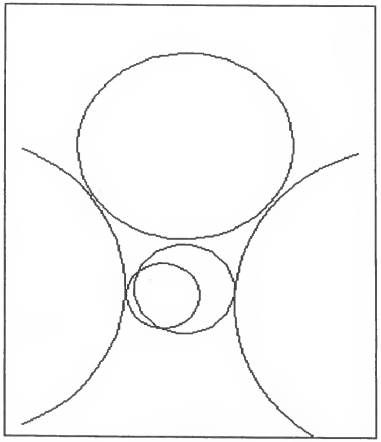
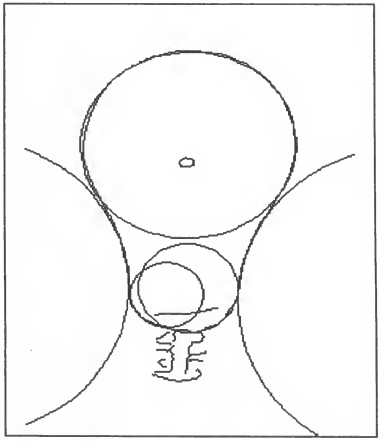


Figure 2.16 Tangent orientation and curvature of the bulb boundary



(a) Detected conic sections



(b) Detected conic sections superimposed on the original image

Figure 2.17 Detected lines and conic sections

CHAPTER 3
REGION-BASED SEGMENTATION

We discussed an edge-based approach to the image segmentation problem in chapter 2. Edge detection focused on the nonhomogeneity between regions. On the contrary, the principal spirit of the region-based approach to the image segmentation problem is the grouping of pixels with the same or similar properties. We want to subdivide an image into regions which have a certain uniformity within each region, that is to say, have the same intensity, color, texture, or same surface characteristics.

The image we want to segment in this dissertation is not the traditional gray level image. It is a surface orientation map (called needle map by some authors) generated by a photometric stereo method. Unlike gray level images, the surface orientation map provides explicit information about the three-dimensional shape of objects in the scene, namely a surface normal vector at each pixel location of the scene. This explicit three-dimensional information must first be organized into meaningful structures (e.g., surface patch description which will be discussed in chapter 4), to understand the scene. The problem we are interested in is the segmentation of the surface orientation map into homogeneous

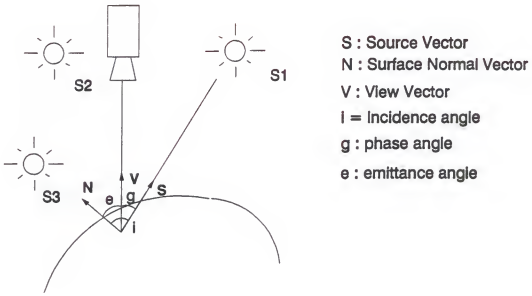
surface patches, where each patch has the same surface characteristic such as mean and Gaussian curvature signs. Since it requires local surface properties such as curvature values, initial processing is performed to compute these values. Once this is done, the segmentation of the surface orientation map becomes the same problem as that of the gray level image segmentation, and it uses the same ideas and the same data structure.

Section 1 discusses the photometric stereo method used to obtain a surface orientation map. Section 2 reviews approaches to the region-based segmentation problem. Section 3 goes into detail about the relationship between the fundamental shape descriptors of differential geometry and the surface curvature signs used to define surface type. Section 4 discusses the computation of Gaussian and mean curvatures from the surface orientation map by local surface fitting and surface segmentation, using ten basic surface types. Section 5 summarizes the procedures to obtain surface type label image from the surface orientation map.

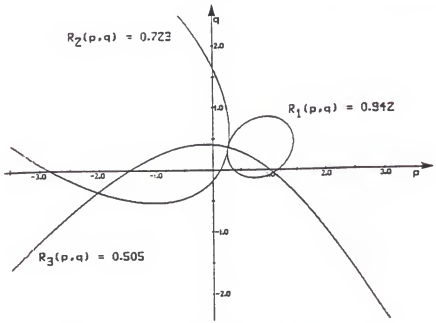
3.1 Generation of a Surface Orientation Map

Image understanding research has produced various techniques for extracting information about the visible surfaces from a scene. Two lines of research that have been investigated extensively are shape from shading [Horn 77] and binocular stereo [Marr and Pogg 79].

The reflectance map [Horn 1977] represents the relationship between surface orientation (a normal vector) and image brightness. Since the direction of a surface normal has two degrees of freedom, we can represent surface orientation by points on a sphere or in a two dimensional plane. The brightness value associated with each surface orientation at a pixel, assuming a fixed light source and viewing configuration, can be obtained either empirically [Wood 80] or analytically from models of the surface microstructure and the surrounding light source arrangement. The photometric stereo method takes multiple images of the same scene, from the same camera position, and with various illumination directions in order to determine surface orientation. Figure 3.1 (a) shows a setup for taking three images for the photometric stereo. Assuming a Lambertian reflection, image brightness is only a function of the light source vector and the surface normal. This setup gives multiple brightness values at each pixel. Since different images are taken from the same point, there is no disparity between the images as there is with binocular stereo, so no correspondence problem has to be solved. Each illumination configuration has a unique reflectance map associated with it; so each of the three brightness measurements is consistent with a different set of surface orientations. Each of these sets corresponds to an iso-brightness on the reflectance map associated with that lighting configuration. The intersection of the three



(a) Imaging geometry for photometric stereo



(b) An example of surface orientation computation

Figure 3.1 Imaging geometry for photometric stereo

contours obtained will yield typically a unique surface orientation at the pixel (see Figure 3.1 (b)). This method is usually implemented by using a lookup table. If we assume that both the viewer and the light source are far from the object, then both the light source direction and the viewer direction are constant over the image. Thus, for a particular light source, the same reflectance map applies everywhere in the image. In practice, a calibration object of known shape is used to determine the relationship between brightness and surface orientation. The points where isobrightness lines intersect can be precalculated and stored as a table of surface orientations that is indexed by triples of brightness values.

But this approach is difficult to use on general surfaces whose reflection varies from specular to near Lambertian. Park and Tou [Park and Tou 90] developed a normal vector equalization method to derive a surface orientation map for such surfaces using three images taken with different locations of light source. The technique is based on the assumption that normal vectors obtained from the Lambertian model are equal to those obtained from the simplified Torrance-Sparrow model of specular reflection. In this dissertation, we employed Park and Tou's method to obtain a surface orientation map (needle map). Figures 3.2 through 3.5 show a set of 3 original images and the resulting needle maps. The result of the photometric stereo method is called a needle

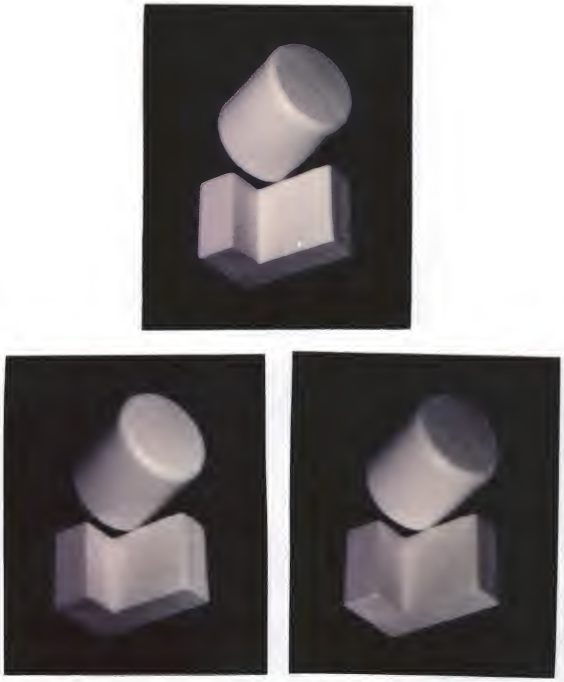


Figure 3.2 Three images of a stair and cylinder

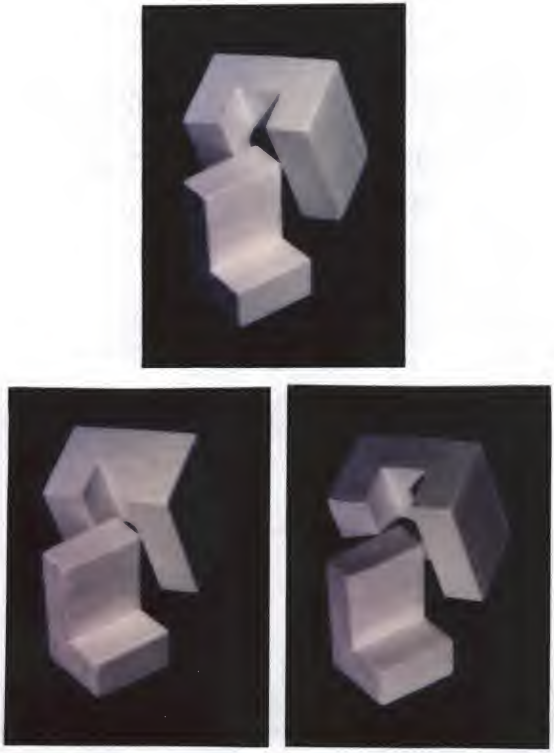
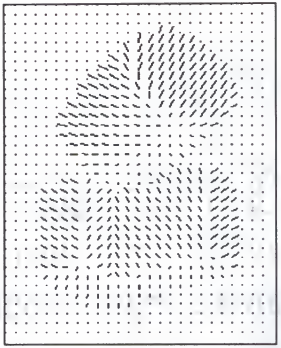
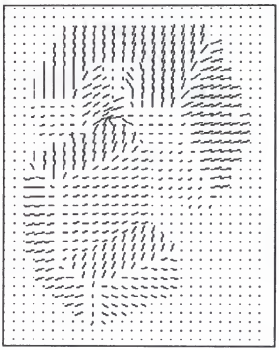


Figure 3.3 Three images of a trough and stair

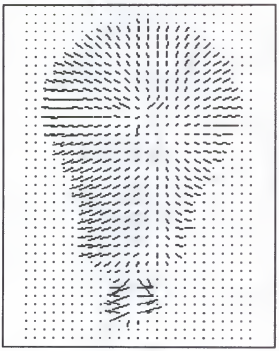
Figure 3.4 Three images of a bulb



(a) Needle map of a stair and cylinder image



(b) Needle map of a trough and stair image



(c) Needle map a bulb image

Figure 3.5 Needle maps generated by the normal vector equalization method

map since it can be shown as a picture of the surface covered with short needles. Each needle is parallel to the local normal. The length of a line, which is the image of one of the needles, depends on how steeply inclined the surface is. The orientation of the line indicates the direction of the steepest descent.

3.2 Review of Region-Based Segmentation

Image segmentation is the process by which an original image (an array of numbers where each entry may represent the gray level, range, or surface orientation) is translated into a description of the regions (the representative properties and the set of addresses of elements within each region).

Let's first define gray level image segmentation in formal terms [Pavl 77].

Let I denote the grid of the sample points of an image, that is to say, the set of integer pairs

$$(i,j) \quad i = 1,2,\ldots,N \ , \ j = 1, 2, \ \ldots \ , M$$

Let f(x,y) be a certain function defined over I. A logical predicate P is defined on subsets S of I as follows:

$$P(S) = \begin{cases} \text{True} & \text{if there exists a constant c such that} \\ & |f(x,y) - c| \leq e \text{ for all points} (x,y) \in S \\ \text{False} & \text{otherwise} \end{cases}$$

where e is a prescribed error tolerance.

Definition 3.1 : A segmentation of I is a partition of I into subsets Si, i=1,...,n for some n such that:

(1) $I = \cup_1^n S_i$

(2) $S_i \cap S_j = \emptyset$ for all $i \neq j$

(3) $P(S_i)$ = True for all i

(4) $P(S_i \cup S_j)$ = False for all $i \neq j$ provided S_i and S_j are adjacent in I.

Condition (1) indicates that the segmentation must be complete to cover an entire image. Condition (2) says regions are disjoint, that is to say, they do not overlap. Condition (3) deals with the properties that must be satisfied by the pixels in a segmented region. All the pixels in the region have the same intensity values, texture properties, or surface properties. Condition (4) indicates that regions S_i and S_j are different in the sense of predicate P. Predicate P may be the same intensity value, texture, color, or surface properties.

The basic concept of region-based image segmentation is clustering the pixels using various measures. There are two main approaches to the region-based image segmentation problem: region growing and split and merge.

The main idea behind the region growing approach is simple: given one or more starting points in the image, adjacent pixels are examined one by one and tested to see if they are close enough to the currently estimated region properties to be accepted. If they fail the test for every attributed region, they will be rejected and a new region is

formed. The motivation behind this approach is very clear; it is an attempt to take both distance in space and similarity of properties into account in the segmentation.

Single linkage region growing schemes [Bric and Fenn 73] regard each pixel as a node in a graph. Neighboring pixels, whose properties are similar enough, are joined by an arc. The image segments are maximal sets of pixels, all belonging to the same connected component. Single linkage image segmentation schemes are attractive for their simplicity. They do, however, have a problem with chaining, because it takes only one arc leaking from one region to a neighboring one to cause the regions to merge. The simplest single linkage scheme defines "similar enough" by pixel differences. Two neighboring pixels are similar enough if the absolute value of the difference between their intensity value is small enough. Hybrid single linkage techniques [Yaki 76] are more powerful than the simple single linkage technique. The hybrid techniques seek to assign a property vector to each pixel, where a property vector depends on the $K \times K$ neighborhood of the pixel. Pixels which are similar are similar because their neighborhoods in some special sense are similar. Similarity is thus established as a function of neighboring pixel values, and this makes the technique better behave on noisy data. In centroid linking region growing, pairs of neighboring pixels are not compared for similarity. The value of a pixel is compared to the mean of an already existing but not

necessarily completed neighboring regions. If its value and the mean value of the region are close enough, then the pixel is added to the region and the mean of the region is updated. If there is more than one region which is close enough, then it is added to the closest region. However, if the means of the two competing regions are close enough, the two regions are merged, and the pixel is added to the merged region. If no neighboring region has its mean close enough, then a new region is established having the given value of the pixel as its first member.

The split and merge method [Horo and Pavl 76, Chen and Pavl 80] for segmentation begins with the entire image as the initial region. Then it successively splits each current region into quarters if the region is not homogeneous enough. Homogeneity can be easily established by determining whether the difference between the largest and the smallest intensities is small enough. The efficiency of the split and merge method can be increased by arbitrarily partitioning the image into square regions of a user selected size, then splitting these further if they are not homogeneous. Because segments are successively divided into quarters, the boundaries produced by the split technique tend to be jagged and slightly artificial.

In conclusion, the difficulty with most of these methods is that both the final solution and the number of iterations to achieve it are highly dependent on the initial conditions.

In general, there is no guarantee that a given image would be segmented in the same way by the same algorithm and two sets of starting points. Another objection is the sequential nature of the search through the image, which requires an enormous computation time compared to edge-based segmentation.

3.3. Local Theory of Surfaces

3.3.1 Importance of Surface Primitives

Surfaces are the features that directly link perception to the objects perceived, and they make information needed to understand and cope with some visual problems explicit(e.g., obscured features). To handle the problem of arbitrary viewing directions, viewpoint invariant surface characteristics are needed that are general enough to describe both polyhedra and objects with arbitrary curved surfaces. Although some special feature recognition approaches offer important advantages for applied computer vision systems, a successful surface matching algorithm for arbitrary surfaces would provide considerably more general object recognition capabilities.

Segmentation of the surface orientation map requires pixels to be grouped together into a relatively small set of symbolic surface primitives. Symbolic surface primitives are characterized by Gaussian and mean curvatures which will be discussed in the next section. The number of symbolic surface primitives should be much smaller than the number of pixels. It should also represent all pixels in the image, and all

pixels should belong to, one and only one, symbolic surface primitive. All pixels within a group of pixels defined by a symbolic surface primitive should be consistent with each other with respect to a defining statement about that symbolic surface primitive.

Transformation of the surface orientation map to symbolic surface primitives requires several internal intermediate levels, where each level uses basic knowledge about surfaces. The first step is to characterize the surface orientation map at every pixel using a small set of surface primitives. In order to do this, Gaussian and mean curvatures are computed in the local neighborhood of every pixel of the surface orientation map. Differential geometry shows that the Gaussian and mean curvatures (equivalently principal curvatures) are fundamental features describing a local surface characteristics.

3.3.2 Functions and Fundamental Forms

We introduce some fundamental notions from differential geometry [Do Carm 76, Hsiu 81].

A regular surface is a set of points $S \subset \mathbb{R}^3$ where, for each $p \in S$, there exists a neighborhood $V \subset \mathbb{R}^3$ and a differentiable homeomorphism $s: U \subset \mathbb{R}^2 \rightarrow S \cap V$, whose differential is one-to-one for each $q \in U$. An element of a regular surface S is a regular point. There are two mathematical entities that are considered in the differential

geometry of smooth surfaces. In classical mathematics of partial derivatives, they are known as the first and second fundamental forms of a surface. Complete knowledge of these forms at every surface point uniquely characterizes and quantifies general smooth surface shape.

We can define a local coordinate frame in which the mapping s takes the form of $\mathbf{s}(x,y) = (x, y, f(x,y))$ for $(x,y) \in U$. U is a parameter space (an image plane) and $\mathbf{s}(x,y)$ is a graph surface defined over U. The first fundamental form measures the small amount of change $\|d\mathbf{s}\|^2$ on the surface at a point (x,y) for a given small vector movement (dx,dy) in the image plane. The first fundamental form I of a surface $\mathbf{s}(x,y)$ is given by the inner product of a differential $d\mathbf{s}$.

$$I = d\mathbf{s} \cdot d\mathbf{s} = E\,dx^2 + 2F\,dxdy + G\,dy^2 = [dx\ dy]\begin{bmatrix} E & F \\ F & G \end{bmatrix}\begin{bmatrix} dx \\ dy \end{bmatrix} \qquad (3.1)$$

$$\text{where } E = (1 + f_x^2),\quad F = f_x \cdot f_y,\quad G = (1 + f_y^2)$$

The second fundamental form measures the correlation between the change in the normal vector $d\mathbf{n}$ and the change in the surface position $d\mathbf{s}$ as a function of a small movement (dx,dy) in the image plane. The second fundamental form II of a surface $\mathbf{s}(x,y)$ is given by

$$II = -\mathbf{ds} \cdot \mathbf{dn} = L\,dx^2 + 2M\,dxdy + N\,dy^2$$

$$= [dx\ dy] \begin{bmatrix} L & M \\ M & N \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \qquad (3.2)$$

where $L = \dfrac{f_{xx}}{\sqrt{1 + f_x^2 + f_y^2}}$, $M = \dfrac{f_{xy}}{\sqrt{1 + f_x^2 + f_y^2}}$, $N = \dfrac{f_{yy}}{\sqrt{1 + f_x^2 + f_y^2}}$

The quantity II/I is known as the normal curvature. Note that normal curvature at a surface point depends on the type of surface and on the direction of the differential vector (dx,dy) in the image plane, equivalently measuring the local curvature along the (dx,dy) direction. A fact of fundamental importance is that the directions of maximum and minimum normal curvatures, called principal directions, are always mutually perpendicular at any point on a smooth surface. The curvatures in these directions are called principal curvatures (k_1, k_2). For example, the two principal directions on a cylinder are parallel and perpendicular to the axis of the cylinder.

The signs of principal curvatures give a qualitative description of a local surface (see Figure 3.6). Gaussian curvature K (the product of two principal curvatures k_1 and k_2) can be used to roughly classify surface types. A point on a surface is called elliptic if $K > 0$, that is to say, two principal curvatures have the same sign. A surface patch consisting entirely of elliptic points is termed locally convex, an example being an egg. When the two principal curvatures have opposite signs then Gaussian curvature K is

negative--the point is called hyperbolic and the surface is locally saddle shaped. We will call a surface patch hyperbolic if it consists entirely of hyperbolic points. When one(or both) of principal curvatures vanish, then K becomes zero and the point is called parabolic. There are two cases. When the mean curvature H (the average of two principal curvatures) is also zero, the point is on a plane. When H ≠ 0 then the point is on a cylinder.

Gaussian curvature K and mean curvature H are computed from E,F,G,L,M,N as follows [Hsiu 81].

$$K = k_1 k_2 = \frac{LN - M^2}{EG - F^2}$$
$$H = \frac{1}{2}(k_1 + k_2) = \frac{1}{2}\,\frac{EN - 2FM + GL}{EG - F^2} \qquad (3.3)$$

The two principal curvatures k_1 and k_2 can be solved from the above equation. Substituting $k_2 = 2H - k_1$ into the first equation of equation (3.3), we have

$$k_1^2 - 2H k_1 + K = 0$$
$$k_1, k_2 = H \pm \sqrt{H^2 - K} \qquad (3.4)$$
$$H^2 - K = \frac{1}{4}(k_1 - k_2)^2 \geq 0$$

The normalized principal direction vectors in the (x,y) image plane for the principal curvatures k_1 and k_2 are given [Besl and Jain 86]

$$\theta_M = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} GM - FN \\ \frac{1}{2}(EN - GL) + (EG - F^2)\sqrt{H^2 - K^2} \end{pmatrix}$$

$$\theta_m = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(EN - GL) + (EG - F^2)\sqrt{H^2 - K^2} \\ FL - EM \end{pmatrix} \qquad (3.5)$$

where E,F,G,L,M,N are defined in eqs. (3.1) and (3.2)

Note that these directions are in general not orthogonal in the (x,y) image plane, even though the 3-D principal curvature directions in the tangent planes at the surface point are orthogonal.

If only the signs of the principal curvatures are used to determine basic surface types, six surface types result: convex ellipsoid, concave ellipsoid, plane, convex cylinder, concave cylinder, and saddle as shown in Table 3.1. The signs of mean and Gaussian curvatures yield eight basic surface types [Besl and Jain 86], as shown in Table 3.2, because saddle surfaces can be resolved into saddle ridge, saddle valley, and minimal surfaces. We used two additional surface types, positive and negative high curvature regions, to make ten surface types. These high curvature regions are introduced to facilitate the segmentation process which will be discussed in the next section. Figure 3.6 shows the shapes of the basic six surface types. The saddle surface type can be further classified as saddle ridge, saddle valley, and

Table 3.1 Surface Types by Principal Curvatures

	k1 < 0	k1 = 0	k1 > 0
k2 < 0	convex ellipsoid	convex cylinder	saddle surface
k2 = 0	convex cylinder	plane	concave cylinder
k2 > 0	saddle surface	concave cylinder	concave ellipsoid

Table 3.2 Surface Types by Mean and Gaussian Curvatures

	K < 0	K = 0	K > 0
H ≤ Next	Negative High Curvature Region		
Next < H < 0	saddle ridge	convex cylinder	convex ellipsoid
H = 0	minimal	plane	none
Pext > H > 0	saddle valley	concave cylinder	concave ellipsoid
H ≥ Pext	Positive High Curvature Region		

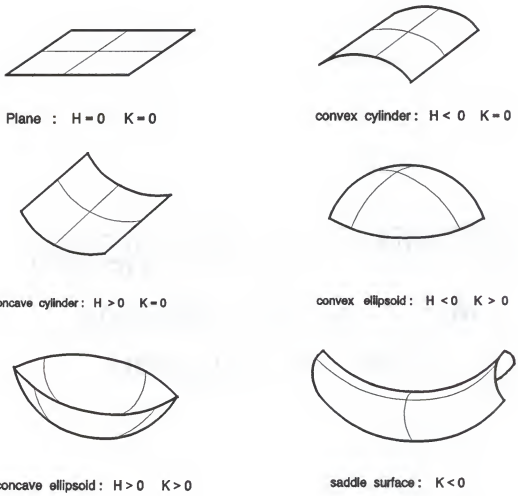


Figure 3.6 Six basic surface types by mean and Gaussian curvatures. Saddle surface type can be further classified as saddle ridge, saddle valley, and minimal.

minimal surface types depending on the mean curvature sign. In this dissertation we use the signs of Gaussian and mean curvatures to classify surface types. There are several advantages using Gaussian and mean curvatures over principal curvatures [Besl 86].

(1) Principal curvature values should be associated with the corresponding principal directions for meaningful interpretation, whereas mean and Gaussian curvature values are direction-free quantities.

(2) If only the signs of the principal curvatures are used to determine basic surface types, six surface types result. The signs of mean and Gaussian curvature yield eight basic surface types.

(3) Gaussian curvature exhibits isometric invariance properties that are not exhibited by either of the principal curvatures. Gaussian curvature is an intrinsic property of a surface. Both principal curvatures and the mean curvature are extrinsic properties of a surface. The intrinsic properties of a surface are not affected by the choice of a coordinate system or viewpoint, whereas the extrinsic properties are affected.

3.4 Curvature Computation from the Surface Orientation Map

The surface orientation map (needle map) generated from the photometric stereo contains too much noise components due to surface irregularity, shadow effect, and inherent noise for

direct use in surface segmentation. Smoothing is needed to filter out local fluctuations in the data due to quantization and measurement noise so as to obtain reasonable differential geometry quantities from the surface orientation map. However, as a result of low-pass filter smoothing, sharp discontinuities in the surface orientation across surface edges are also blurred. We wish to preserve the discontinuities to locate the edges. In order to achieve reasonable smoothing, we need a relatively large smoothing kernel. Blind application of a large smoothing kernel blurs the boundary and spills unwanted information across distinct surface boundaries.

In order to preserve the boundary, the needle map is first smoothed by the repeated application of a Gaussian filter with a small size smoothing kernel. If the smoothing window is overlapped with an edge pixel, smoothing is not applied over that window. From the Central Limit Theorem, repeated application of small window can achieve the equivalent function of a large window. The smoothed surface orientation map is then normalized for curvature computation.

The unit surface normal vector and the first partial derivatives of a surface $z = f(x,y)$ are related in the following way. The gradient vector of $F(x,y,z) = f(x,y) - z = 0$ is normal to the surface $F(x,y,z) = 0$

$$\text{Grad } F(x,y,z) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right) = (f_x, f_y, -1) \qquad (3.6)$$

By normalizing grad F, we obtain a unit normal vector

$$(N_x, N_y, N_z) = \left(\frac{f_x}{\sqrt{f_x^2+f_y^2+1}}, \frac{f_y}{\sqrt{f_x^2+f_y^2+1}}, \frac{-1}{\sqrt{f_x^2+f_y^2+1}}\right) \qquad (3.7)$$

The first partial derivatives f_x, f_y are obtained from a unit surface normal

$$f_x = -\frac{N_x}{N_z}, \quad f_y = -\frac{N_y}{N_z} \qquad (3.8)$$

Gaussian and mean curvatures can be obtained by computing the second partial derivatives f_{xx}, f_{xy}, f_{yy} from f_x, f_y and using (3.3). But this method does not provide reliable curvature values.

For our purpose, a local least squares surface fit is computed within the N × N window around each pixel where N is odd. Surface patches are estimated initially under the assumption that locally the underlying surface can be approximated by some surface equation. There are many choices for modeling a local surface fit, including a polynomial approximation, orthogonal polynomial approximation, or spline approximation. We will use third order cubic surface approximation. A local surface patch is approximated by

$$f(x,y) = a_1x^3 + a_2y^3 + a_3x^2y + a_4xy^2 + a_5x^2 + a_6y^2 + a_7xy + a_8x + a_9y + a_{10} \quad (3.9)$$

We want to estimate the bicubic parameters $\mathbf{A} = [a_1, a_2, \ldots, a_9]^T$. Since we are using a needle map which provides only a normal vector at each pixel, we can not solve for the displacement term a_{10}. The partial derivatives of Eq. (3.9) in x,y directions are

$$\begin{aligned} f_x &= 3\cdot a_1x^2 + 2\cdot a_3xy + a_4y^2 + 2\cdot a_5x + a_7y + a_8 \\ f_y &= 3\cdot a_2y^2 + a_3x^2 + 2\cdot a_4xy + 2\cdot a_6y + a_7x + a_9 \end{aligned} \quad (3.10)$$

For each point involved in the approximation, we are given 2 equations. If n points are involved in the approximation, we have the following 2n simultaneous equations. This set of equations is a description of the overconstrained system around the window.

$$\begin{aligned} f_{x1} &= 3\cdot a_1x_1^2 + 2\cdot a_3x_1y_1 + a_4y_1^2 + 2\cdot a_5x_1 + a_7y_1 + a_8 \\ f_{y1} &= 3\cdot a_2y_1^2 + a_3x_1^2 + 2\cdot a_4x_1y_1 + 2\cdot a_6y_1 + a_7x_1 + a_9 \\ f_{x2} &= 3\cdot a_1x_2^2 + 2\cdot a_3x_2y_2 + a_4y_2^2 + 2\cdot a_5y_2 + a_7y_2 + a_8 \\ f_{y2} &= 3\cdot a_2y_2^2 + a_3x_2^2 + 2\cdot a_4x_2y_2 + 2\cdot a_6y_2 + a_7x_2 + a_9 \\ &\quad \cdots\cdots\cdots\cdots\cdots\cdots \\ f_{xn} &= 3\cdot a_1x_n^2 + 2\cdot a_3x_ny_n + a_4y_n^2 + 2\cdot a_5x_n + a_7y_n + a_8 \\ f_{yn} &= 3\cdot a_2y_n^2 + a_3x_n^2 + 2\cdot a_4x_ny_n + 2\cdot a_6y_n + a_7x_n + a_9 \end{aligned} \quad (3.11)$$

In terms of matrix notation, the above equation can be written as

$$\begin{bmatrix} 3x_1^2 & 0 & 2x_1y_1 & y_1^2 & 2x_1 & 0 & y_1 & 1 & 0 \\ 0 & 3y_1^2 & x_1^2 & 2x_1y_1 & 0 & 2y_1 & x_1 & 0 & 1 \\ 3x_2^2 & 0 & 2x_2y_2 & y_2^2 & 2x_2 & 0 & y_2 & 1 & 0 \\ 0 & 3y_2^2 & x_2^2 & 2x_2y_2 & 0 & 2y_2 & x_2 & 0 & 1 \\ \cdot & \cdot & & & & & & & \\ 3x_n^2 & 0 & 2x_ny_n & y_n^2 & 2x_n & 0 & y_n & 1 & 0 \\ 0 & 3y_n^2 & x_n^2 & 2x_ny_n & 0 & 2y_n & x_n & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ \cdot \\ a_8 \\ a_9 \end{bmatrix} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ \cdot \\ f_{xn} \\ f_{yn} \end{bmatrix} \qquad (3.12)$$

$$\mathbf{X}\ \mathbf{A} = \mathbf{F}$$

where matrix $\mathbf{X}$ is computed from the selected window, $\mathbf{A}$ is the coefficient vector of the fitted surface, and $\mathbf{F}$ is a column vector of partial derivatives which can be computed from the surface normal vectors within the window. For a $N \times N$ window, we have $n = 2N^2$ number of equations. The least squares normal equations are formed by multiplying through by the transpose of the $\mathbf{X}$ matrix.

$$\mathbf{X}^T\ \mathbf{X}\ \mathbf{A} = \mathbf{X}^T\ \mathbf{F} \qquad (3.13)$$

Now the normal equation represents a square 9 by 9 matrix equation. The coefficient vector is solved analytically as follows.

$$\mathbf{A} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{F} \tag{3.14}$$

If the square matrix of the normal equations is singular, the singular value decomposition can be used to return a coefficient vector which fits the degenerate data. For a given window size, $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ can be precomputed to speed up the computation of the coefficient vector.

We evaluate $f_x, f_y, f_{xx}, f_{xy}, f_{yy}$ at the window center (0,0), which result in very simple forms. From equation (3.10)

$$\begin{aligned} f_x(0,0) &= a_8 \\ f_y(0,0) &= a_9 \\ f_{xx}(0,0) &= 2 \cdot a_5 \\ f_{xy}(0,0) &= a_7 \\ f_{yy}(0,0) &= 2 \cdot a_6 \end{aligned} \tag{3.15}$$

The Gaussian curvature(K), the mean curvature(H), the principal curvatures (k_1, k_2), and the principal directions (θ_M, θ_m) are then computed using the equations (3.3), (3.4), and (3.5).

Each pixel is then assigned to one of the ten surface types according to the signs of the mean and Gaussian curvature values. We introduce the concept of high curvature regions to facilitate the segmentation. These high curvature regions prevent the same surface types with different surface orientation from being labeled as the same region. A pixel is assigned to a positive high curvature region type if the mean curvature is above a threshold. Similarly, a negative high

curvature region type is assigned to a pixel if the mean curvature is below another threshold. The threshold value is not critical, because these high curvature regions will eventually be removed. These high curvature regions play crucial roles in the integration phase, which registers the edge detection output and the surface property based segmentation output.

Since we are dealing with noisy data, exact zero value of these two curvatures is difficult to obtain. A toleranced signum function

$$\mathrm{sgn}_{\epsilon}(x) = \begin{cases} +1 \text{ if } x > \epsilon_1 \\ 0 \text{ if } \epsilon_2 \leq x \leq \epsilon_1 \\ -1 \text{ if } x < \epsilon_2 \end{cases} \tag{3.16}$$

is used to compute the individual surface curvature signs. A signed mean curvature image (sgn(H)) is generated using the selected thresholds, where a pixel with mean curvature greater than a positive high curvature threshold is labeled 2, and a pixel with mean curvature less than a negative high curvature threshold is labeled -2. A pixel with mean curvatures between these two thresholds is labeled using (3.16). The signed Gaussian curvature image (sgn(K)) is generated using (3.16). The sgn(H) and sgn(K) are then used to determine the surface type label image, where each pixel is assigned a surface type label by the following procedure.

```
If signed mean curvature label is -2
   Assign label 10 to the pixel
else if signed mean curvature label is 2
   Assign label 12 to the pixel
else
   Assign the label determined by the following formula
      1 + 3(1 + sgn(H)) + (1 - sgn(K))
```

For example, if a pixel is a convex cylinder surface type, its label in the signed mean curvature image (sgn(H)) is -1 and its label in the signed Gaussian curvature image (sgn(K)) is 0, hence surface type label for convex cylinder is 2 by the above formula.

The initial segmentation result usually contains many small isolated regions due to the surface irregularity and lighting conditions. The small noisy regions are removed by shrinking and expanding operations. Shrinking reduces each region size, maintaining its gross shape. Each pixel is checked with a 3×3 window to see if any of its 8-neighbors is different from the center pixel. If there is any neighbor which is different from the center pixel, then the center pixel is a border pixel and it is marked to be filled by the expansion algorithm. Small regions will disappear and a special label will be stored in its place. Large regions will remain and eventually be expanded to take over the marked pixels by the shrinking process.

After small noisy regions are removed, high curvature regions are allowed to expand several pixels into the other regions. There are two reasons for performing this operation. The first reason is that edges generated from the edge detection channel may not be correctly registered with the high curvature regions because the photometric stereo method can not provide accurate surface orientation information at the boundaries. Expansion of the high curvature regions allows more reliable registration of the edges with the surface property based segmentation result. The second reason is that, sometimes, high curvature regions may not completely separate two different regions of the same surface type. This is true at vertices where surface normal vectors generated from the photometric stereo method behave irregularly due to the mutual illumination effect. By expanding the high curvature regions we can improve region separation.

Figure 3.7 is a data flow diagram to obtain a surface segmentation from a surface orientation map. Figures 3.8 (a), 3.9 (a), and 3.10 (a) are the initial surface segmentation results of three surface orientation maps. A cylinder and stair, a stair and trough, and a bulb have been segmented using the extended Gaussian and mean curvature signs. Figures 3.8 (b), 3.9 (b), and 3.10 (b) are the refined segmentation results where very small noisy regions are removed by shrinking and expansion operations. High curvature regions are then expanded by two pixels. The noise removed

segmentation output, together with the edge detection output, will be input to the integration module where interpretation of edges and surfaces starts.

3.5 Summary

In this chapter we used viewpoint invariant surface characterization using Gaussian and mean curvature signs. The signs of Gaussian and mean curvatures classify surfaces into eight basic surface types. We introduced two additional surface types, positive high curvature region and negative high curvature region, to facilitate the segmentation by preventing the same surface types with different surface orientation from being labeled as the same region. Since high curvature regions usually correspond to the edges of objects, these high curvature regions play a crucial role in the integration phase of edge based and surface based segmentation. Orientation discontinuity preserving smoothing is first performed on the surface orientation map. Gaussian and mean curvatures are reliably computed through local surface fitting on the smoothed surface orientation map. Initial surface-based segmentation is obtained using the signs of Gaussian and mean curvatures. Small noisy regions are removed by shrinking and expansion operation. Segmentation is further refined by the expansion of high curvature regions.

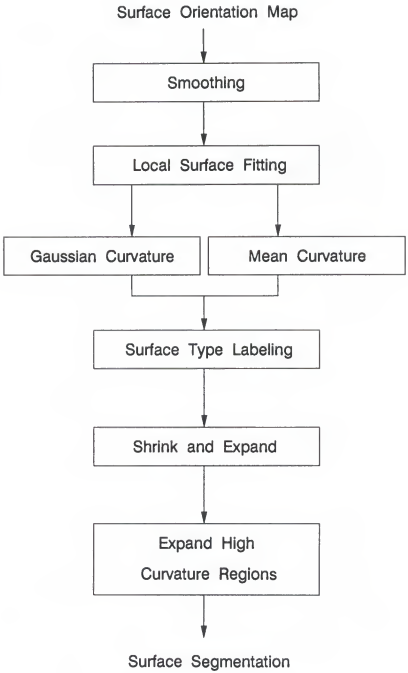
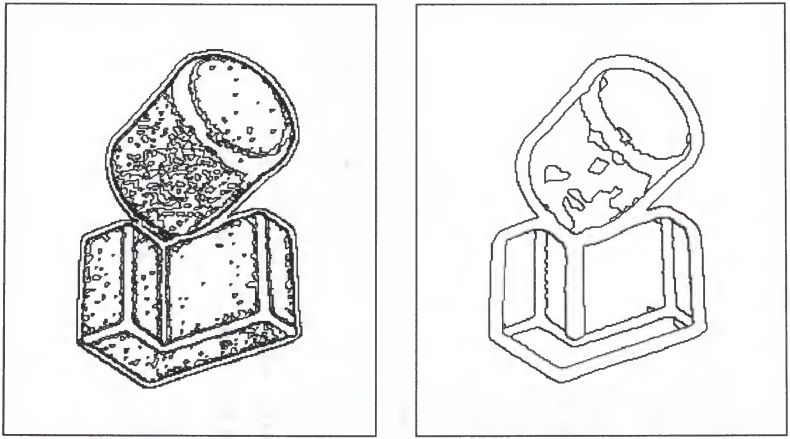


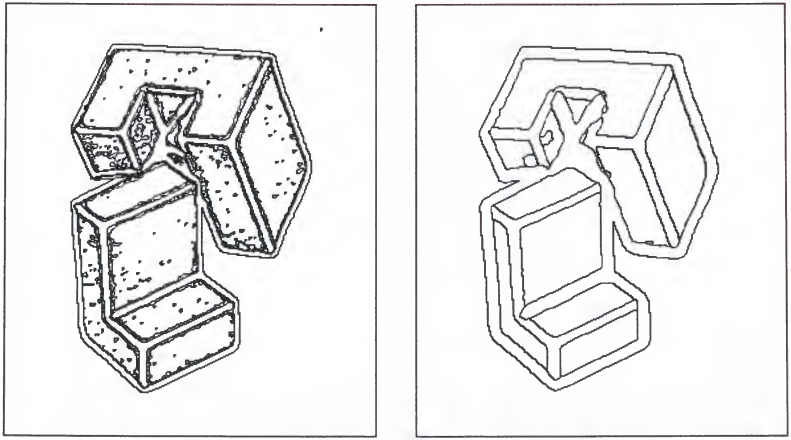
Figure 3.7 Data flow diagram of surface segmentation



(a) Initial segmentation by extended HK sign

(b) Segmentation after shrink and expand

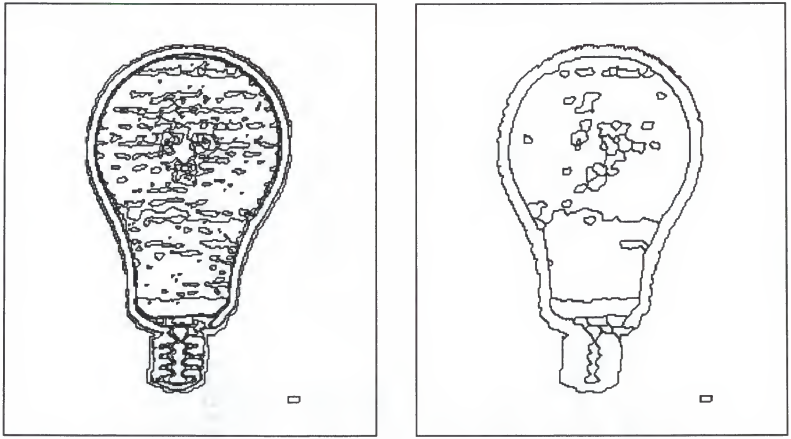
Figure 3.8 Surface segmentation of a cylinder and stair image by extended Gaussian and mean curvature signs



(a) Initial segmentation by extended HK sign

(b) Segmentation after shrink and expand

Figure 3.9 Surface segmentation of a stair and trough image by extended Gaussian and mean curvature signs



(a) Initial segmentation by extended HK sign

(b) Segmentation after shrink and expand

Figure 3.10 Surface segmentation of a bulb image by extended Gaussian and mean curvature signs

CHAPTER 4
INTEGRATION

4.1 Introduction

In the previous two chapters, we discussed two different approaches to the image segmentation problem in detail. Ideally, either approach should produce a complete segmentation of a given image where each region would correspond to the objects, surfaces, or parts of objects of the scene to be interpreted. However, low level processes can only produce partitions on a nonsemantic basis, since low level operations are inherently based on local information.

Edge detection output rarely describes complete regions. There are inevitable gaps around junctions, missing edges, and spurious edges. On the other hand, the result of region based segmentation is usually oversegmented or undersegmented, and the boundaries of regions may not correspond to the boundaries of objects in the scene.

The results of these low level operations should be looked at with a more global perspective in order to compensate for errors incurred in low level processes. The role of the integration is to fuse the two imperfect outputs from these two independent channels into a coherent and consistent segmentation, so that recognition can start with

more reliable information. The reliable structured information will enable the recognition process to prune the enormous search space of model data base at an early stage.

4.2 Region Adjacency Graph

In section 3.4, we computed the mean(H) and Gaussian(K) curvature values at each pixel of the surface orientation map by local bicubic surface fitting. These curvature values, together with their signs, assign one of the basic ten surface types to each pixel. Initial noise removal was performed by a shrinking and expansion operation. The surface type label image thus obtained is partitions of an image, where each partition contains pixels of the same surface type label. The surface type label image is usually an incomplete segmentation of the given image, and region merging and splitting may be required.

For region operations such as region merging and splitting, each connected partition (region) must be treated and accessed as a different entity, hence must be assigned a different region number. For example, two adjacent planar regions of different orientations in the surface type label image have the same surface type label but must be labeled as different regions for region operations. We have to find all the four-connected regions in the surface type label image. A four-connected region has the property that any two pixels

in the region can be connected by a string of four-connected pixels all belonging to the same region.

A connected component labeling algorithm in Figure 4.1 finds all the four-connected regions and assigns a different region number to each four-connected region in the surface type label image. For example, in Figure 4.2, R9 and R14 are plane surface type in surface type label image (all the pixels in R9 and R14 have the same plane surface type label 5). After connected component analysis, pixels in R9 and R14 have different region numbers(9 for R9 and 14 for R14) because R9 and R14 are not connected, hence can be accessed independently.

The next step is to build a region adjacency graph (RAG) from the output of the connected component analysis to make the adjacency and inclusion relationship between regions explicit. A region adjacency graph is two-tuple (V,E) where each node in V corresponds to a region, and an arc in E joining two nodes represents that two regions are adjacent. A RAG is constructed by tracing each region boundary and finding the adjacent regions on the boundary. During the region boundary tracing, we also record the degree of adjacency between two regions, which is the length of the common boundary between two regions. Degrees of adjacency between regions are used to decide which two regions should be merged. Figure 4.2 is an example of the surface segmentation of a stair and cylinder image and its corresponding region

```
Algorithm   Connected Component Labeling

Input :   Surface type label image  S(i,j) of size M × M
Output :  Connected component image  C(i,j) of size M × M
          where each region has different region number


BEGIN
     NR = 1 ;  { NR is the region number }
     FOR i=2 TO M-1 ;
          FOR j=2 TO M-1 ;
               IF  S(i,j)=S(i-1,j)  AND  S(i,j)≠S(i,j-1)
                    THEN  C(i,j) = C(i-1,j)
               ELSE IF  S(i,j)=S(i,j-1)  AND  S(i,j)≠S(i-1,j)
                    THEN  C(i,j) = C(i,j-1)
               ELSE IF  S(i,j)≠S(i,j-1)  AND  S(i,j)≠S(i-1,j)
                    THEN  C(i,j) = NR   { new region }
                          NR = NR + 1
               ELSE
                          C(i,j) = C(i,j-1)
                          Record C(i-1,j) is equivalent to
                               C(i,j-1)
          { end FOR }
     { end FOR }

     Use the equivalence table to assign unique region number
     to equivalent pixels
END
```

Figure 4.1 Connected component algorithm

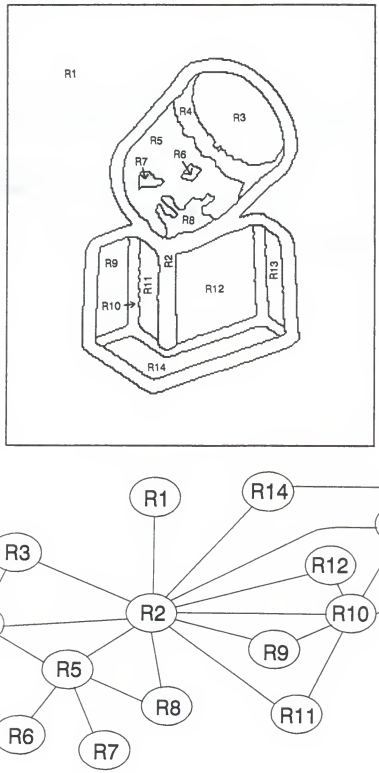


Figure 4.2 Surface segmentation and the region adjacency graph

adjacency graph. According to the adjacency property of regions a RAG is always planar, which means no arc intersects another arc. The important properties of a RAG are:

1) The degree of a node (the number of arcs connecting to other nodes) corresponds to the number of regions adjacent to that region. In Figure 4.2 node (region) R10 has degree 6, denoting it is adjacent to 6 other regions.
2) If a region surrounds other regions completely, then the corresponding node is a cutnode in the RAG. In Figure 4.2, node R5 is a cut node because R6 and R7 have no arcs connecting to other nodes if R5 is removed. If none of the regions in the image enclose other regions, the RAG will have no cutnode.

In addition to the topological properties of region adjacency and enclosure, nodes of a RAG should store geometrical information about the regions such as area, centroid, perimeter, surface type, and compact ratio so that region merging and splitting operations can use these information. The geometrical and adjacency information of regions in the final segmentation result are also used by the recognition module through surface patch matching and through the computation of the viewing transformation. The region frame in Figure 4.3 contains the geometrical and adjacency information about the region. Each slot of the region frame is explained below.

1) Enclosing_rectangle: In order to facilitate the access of each region, the smallest rectangular region completely enclosing the region is associated with each region. The coordinates of the upper left corner and the lower right corner of the rectangle are attached to the enclosing rectangle slot of the region. When a region needs to be accessed, only the enclosing rectangle is examined instead of the whole image.
2) Surface_type: This slot contains the surface type of the region in surface segmentation. The surface type is one of the ten surface types defined in Table 3.2.
3) Bounding_edge: This slot contains the list of edge segments associated with the region after integration of surface segmentation and edge segments.
4) Angle_list: This slot contains the absolute angle list of the polygon of planar surface type region. For a nonpolygon type surface this slot points to empty list.
5) Surface_orientation: This slot contains the surface normal information of the planar surface after surface fitting. This slot is empty for nonplanar surface types.
6) Area: This slot contains the number of pixels in the region.
7) Perimeter: The perimeter of a region is computed as the boundary of the region is traversed:

(Number of even chain codes)+$\sqrt{2}$*(Number of odd chain codes)

8) Compact ratio: The compact ratio of a region is given as the ratio of the square of the perimeter to the area.

9) Centroid: The centroid of a region is given as the average of each coordinate of pixels in the region:

$$x_c = (\Sigma\ x_i)/N \qquad y_c = (\Sigma\ y_i)/N$$

where N is the number of pixels in a region (area).

10) Adjacent_Region: This slot contains a list of adjacent regions.

```
(Region_23  (Enclosing_Rectangle (x1, y1) (x2, y2))
            (Surface_type ( Plane, convex ellipsoid,..))
            (Bounding_edge ( e1 e2 ...))
            (Angle_list  ( Δe1e2  Δe2e3 ...))
            (Surface_orientation   ( Nx Ny Nz ))
            (Area   246)
            (Perimeter  447)
            (Compact_ratio   2.54)
            (Centroid  (x3 y3))
            (Adjacent_Region (R21  R17  R56 ...)))
```

Figure 4.3 An example of a region frame

Bounding_edge, Angle_list, and Surface_orientation slots are filled after edge segments labeling and surface fitting.

4.3 Line Labeling

Edges in the image along with the surfaces enclosed by the edges provide essential information for object recognition. Edge segments in an image have different physical meanings in the scene. In the following discussion edges and regions in Figure 4.4 are used as examples.

(1) a depth discontinuity - One surface occludes the other surface (edges e1, e2, and e4).

(2) a surface orientation discontinuity - Surface normal discontinuity occurs at the edge (edges e2, e3, and e6).

(3) a reflection discontinuity - Reflection characteristic changes on the same surface (e.g., surface marking) (edge e7).

(4) an illumination discontinuity - This occurs at the shadow boundary (edge e5 in Figure 4.4).

In this dissertation only depth and surface orientation discontinuity are considered. Each edge segment can be classified as either the projection of a limb or an edge. A limb is the locus of points on the surface where the line of sight is tangent to the surface (edges e1 and e4). An edge is a tangent plane discontinuity (edges e2, e3, e6). Additionally, each edge can be classified as a convex (e3), concave (e6), or occluding edge (e2). For occluding edges and limbs we would like to infer which of the two surfaces bordering the edge segment in the image is nearer to the viewer. Since surface R2 occludes surface R1, it is nearer to

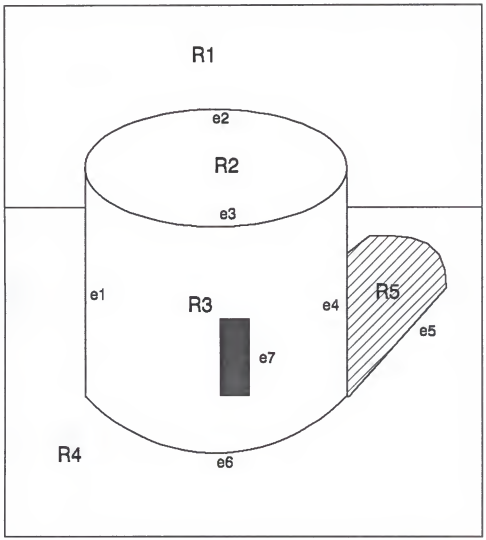


Figure 4.4 Edges in an image

the viewer. Surface R3 containing limb e1 is closer to the viewer than surface R4 at the limb. These inferences can be represented by labeling each edge segment with one out of 6 possible labels.

1. A "+" label represents a convex edge - an orientation discontinuity such that two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle greater than π in 3-D.
2. A "-" label represents a concave edge - an orientation discontinuity such that two surfaces meeting along the edge in the scene enclose a filled volume corresponding to a dihedral angle less than π in 3-D.
3. A "←" or a "→" represents an occluding convex edge. When viewed from the camera, both surface patches which meet along the edge lie on the same side, one occluding the other. As one moves in the direction of the arrow, the surfaces are to the right.
4. A "←←" or a "→→" represents a limb. Here the surface curves smoothly around to occlude itself. As one moves in the direction of the twin arrows, the surface lies to the right. The line of sight is tangential to the surface for all points on the limb. Limbs move on the surface of the object as the viewpoint changes.

Huffman [Huff 71] enumerated all of the permissible junctions in line drawings in the domain of trihedral polyhedra, and used them for categorizing line segments

according to their physical types as described above. Waltz [Walt 76] extended the Huffman method to drawings with shadows and cracks. Malik [Malik 85] expanded the method to curved objects. Malik's junction dictionary is shown in Figure 4.5.

The junctions, intersections of lines in an image, are categorized into 7 types. L, Arrow, and Y junctions, which are coterminations of lines in a 2-D image, are projections of vertices that are the cotermination of edges in 3-D. A Y-junction is the common intersection of three lines, and the angles between two neighboring lines are less than 180°. An arrow junction is formed by the intersection of three lines, if the largest angle between the neighboring lines is greater than 180°. An L junction is a projection of a vertex, when only one surface containing the vertex is visible.

A T junction is a pseudo junction in that it is not a physical projection of a vertex. Instead, it occurs when one surface occludes another surface such that only part of the occluded surface is visible. The surface which contains the shaft of the T as one of its boundaries is farther from a viewer than the surface which contains the top of the T. Thus the T junction plays a central role in separating different objects in an image. A curve ends at a terminal junction (see Figure 4.6). At a curvature L junction, the tangent is continuous but the curvature is discontinuous. At a three-tangent junction, three curves share a common tangent and two

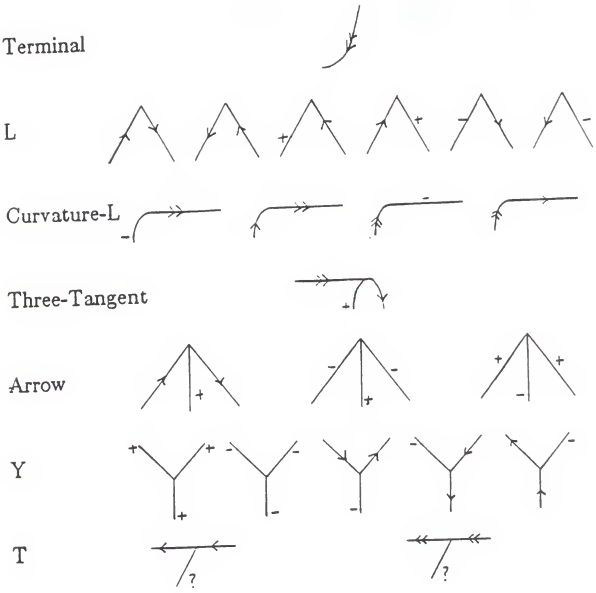


Figure 4.5 Junction Dictionary (from Malik 87)

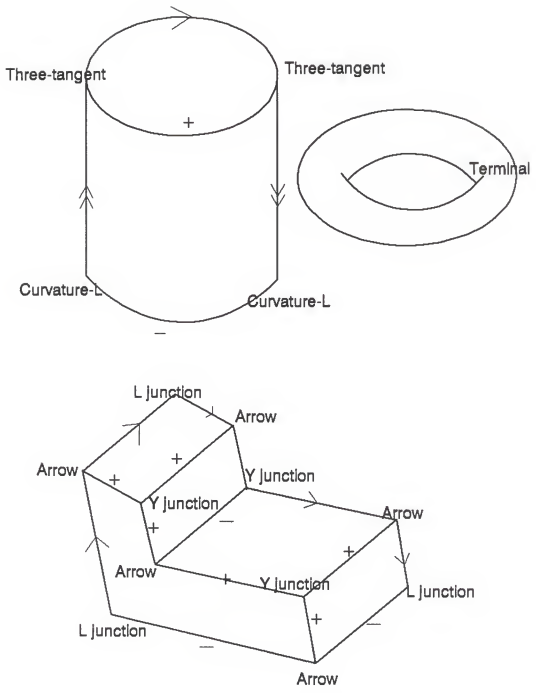


Figure 4.6 Examples of line and junction labels

curves have the same curvature. Figure 4.6 shows an example of line labeling.

Unlike range images, we can not detect a jump boundary (depth discontinuity) in a surface orientation map. Only orientation discontinuity edges are detected. The mean curvature is large at the orientation discontinuity edge. In section 3.4, we defined two surface types, positive high curvature region and negative high curvature region, where the mean curvature is above or below a given threshold. An edge segment in a negative high curvature region corresponds to a convex edge, and it signifies that two surfaces joined by the edge belong to the same object. The interpretation of an edge segment in positive high curvature region is a little complicated in our case. It may correspond either to a real concave edge of the same object or to an occluding boundary separating two objects. An occluding boundary in turn may be a surface orientation discontinuity or a limb where surface normals orthogonal to the viewing direction turn away from the viewer. Fortunately, the surface type associated with the edge can decide the nature of the occluding boundary, that is to say, an occluding edge of the plane region is orientation discontinuity, and an occluding edge of all the other surface types is depth discontinuity. Final verdict on the meaning of a concave edge should be decided using the more global information to be discussed later.

The line labeling alone does not convey much information, and it is only a necessary condition for a line drawing to be physically realizable. The main difference of line labeling usage in this dissertation from previous approaches is that we will use the junction dictionary mainly as a prediction and confirmation tool for surface and edge interpretation. This is possible because convexity or concavity of an edge can be initially determined by its location in the surface based segmentation output.

4.4 Integration of Surface-Based and Edge-Based Segmentation

Now we can perform a region analysis using the region adjacency graph. First, we have to clean some of the remaining small regions adjacent to a large region and artifact regions created by our segmentation method. These small regions can be easily merged with the large adjacent regions without any semantic consideration because their sizes reflect noisiness, and merging with the large region does not much affect the properties of the large region. Another type of region to be removed is narrow strip-like regions (usually small convex or concave cylindrical regions) adjacent to a high curvature region whose compact ratios are greater than a prespecified threshold (25). We will call such a region a transient region. At the transient region, surface orientation is in the middle of settling to another value. This usually occurs at the edges or boundaries of objects where curvature values

changes from extreme to near zero due to orientation discontinuities. A transient region should be merged with an adjacent non-high curvature region which shares the longest border with it. A non-high curvature region is one of the eight surface type region which is not a high curvature region defined in section 3.4. We first apply the following rules to the regions created by connected components labeling to obtain a refined surface segmentation.

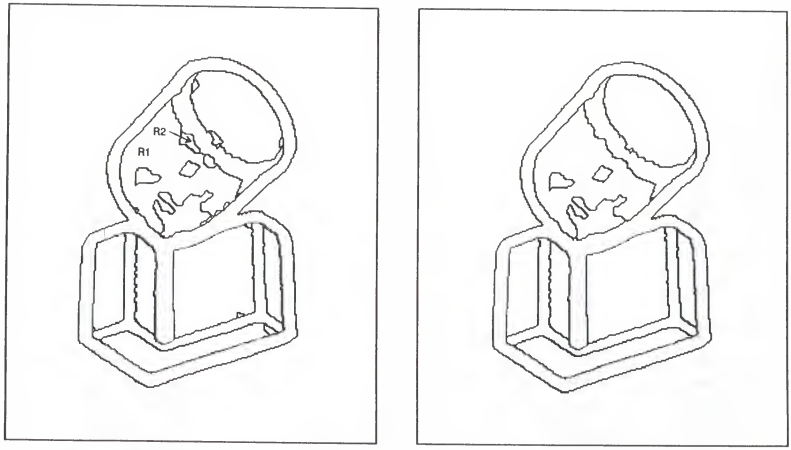
Rule_1

IF (1) a region size is small (less than 60 pixels)
THEN merge with the non-high curvature region with which it shares the longest border

RULE_2

IF (1) a region neighbor is high curvature region
(2) a region compact ratio is high (greater than 25)
THEN merge with an adjacent non-high curvature region

Figure 4.7 is an example of small region merging. Region R2 is merged with R1 using RULE_2. The rest of the small regions are merged with the adjacent non-high curvature regions by RULE_1.



(a) Segmentation after shrinking and expansion

(b) Small region merging

Figure 4.7 An example of small region merging

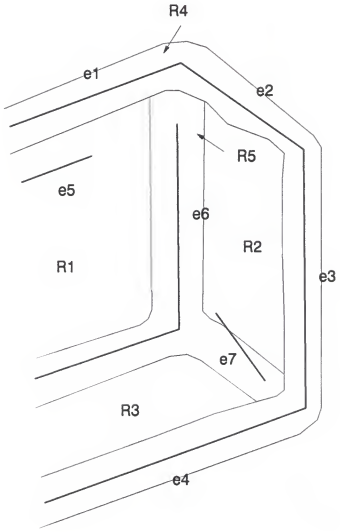
4.4.1 Edge Type Labeling

Now we integrate the edge-based segmentation output and the surface-based segmentation output together. The refined surface segmentation is put into registration with the edge segments produced by the edge segmentation procedures of section 2.4. By examining the region where an edge segment lay we can assign physical meaning to the edge segment. For example, if an edge segment is inside a negative high curvature region, it is labeled as a convex edge. Likewise an edge segment inside a positive high curvature region is labeled as a concave edge. This is the case that the consistent evidences (that is to say, an image edge is a real physical edge) from both channels of segmentation processes reinforce each other. The edge segment labeling algorithm in Figure 4.8 labels edge segments by associating each edge segment with a region and by identifying the region types.

Note that the second steps in the if and else if statements are the superimposition of the labeled edge segments with the surface-based segmentation output for the removal of high curvature regions at the next processing step. For example in Figure 4.9, edge segments e1, e2, e3, and e4 are in positive high curvature region R4, so they are labeled as concave edges and are superimposed on surface-based segmentation. Since edge segment e5 is not inside a high curvature region, it is not superimposed. Since the pixels of an edge segment may be in several regions, an edge segment is

```
Algorithm  Edge Segments Labeling
BEGIN
     FOR each edge segment  {
          WHILE traversing along the edge segment {
               Increment the region number each pixel is in.
          }  /* end while */
          Attach the region number which received the highest
                    vote to the edge segment.
          IF the region is a negative high curvature region
               Update edge label slot with convex edge.
               Put the negative edge segment number into the
                    segmented image.
          ELSE IF the region is a positive high curvature
                    region
               Update edge label slot with concave edge.
               Put the negative edge segment number into the
                    segmented image.
     }  /* end FOR */
END.
```

Figure 4.8 Algorithm for edge segment labeling



R4 : Positive High curvature Region

R5 : Negative High Curvature Region

R1, R2, R3 : Plane region

e1, e2, e3, e4, e5, e6, e7 :

Edge segments from edge contour segmentation

Figure 4.9 Superimposition of edge segments with the surface-based segmentation

considered to be inside the region where the largest number of pixels are in (while loop in segment labeling algorithm). Edge segment e7 is considered to be inside the negative high curvature region R5 because most of pixels in e7 are R5. Therefore, e7 is labeled as a convex edge and is superimposed on surface-based segmentation.

Edge segments which are not assigned a convex or a concave label either are noisy or are possibly a region segmenting line. These edge segments are discarded or used for splitting regions at later stages depending on their relationships with other edges and regions. But an edge segment whose connected neighbor is already assigned a label is retained to maintain continuity.

Now, high curvature regions are removed. High curvature regions are introduced to prevent the surface segmentation process to cross over possible region boundaries and to make edge segment labeling easy. They are artifacts and not real surface patches. High curvature regions are removed by allowing adjacent non-high curvature regions to expand into the high curvature regions. This is easily accomplished by examining the 8-neighborhood of the boundary pixels of high curvature regions. If one of its 8-neighbor pixels is a non-high curvature pixel, then we expand the pixel label into the high curvature region. This erosion of the high curvature regions is repeated until the expansion reaches edge points or encounters other non-high curvature regions. The removal of

high curvature regions results in a segmented image where each region is bounded by the edge segments from the edge detection, or new region boundaries are formed.

Even though edge segments are labeled and superimposed with the surface-based segmentation output, there are no direct links between edge segments and regions. Association of regions and edge segments are achieved by traversing the region boundaries once again. During the boundary traversal, we also generate the new region adjacency graph because some regions are expanded during the removal of high curvature regions. After this traversal, the region slot of each edge segment is updated with the regions which borders the edge segment. Likewise, the edge slot of each region is also updated with the edge segments encountered. After this association, we can easily access the regions associated with each edge segment by simply examining the region slot of the edge segment which is a list of pointers to the regions associated with the edge. Similarly, the edge slot of a region contains a list of pointers to the edges associated with the region.

Since regions and their associated edge segments are known, we can connect broken edge segments and find junctions reliably using region information. First, we examine the list of open edge segments generated in section 2.4. An edge segment is open if one or both endpoints are not connected to another edge segment. For each open edge segment we can

select the associated region. Two open edge segments which borders the same region are combined into one segment if their endpoints are close (the distance is less than 30 pixels) and the difference of their tangent directions is less than 8 degrees. Since we have the region information associated with the edge segments, we can relax the closeness requirement of two edge segments. This enables two edge segments with a long gap between them to connect to each other. This is another case of an advantage of the integrated approach. Edge segment connection is done, supported by the region of which the two open edge segments are boundary.

For edge segments of similar tangent directions (difference is less than 8 degrees) which do not share a common region, we connect them if the distance between endpoints is less than 15 pixels. These conditions are captured in the following rules.

RULE_11

IF (1) the orientation difference of two open edge segments is less than eight degrees
(2) they have the same region on their sides
(3) the distance between endpoints are less than a threshold 1 (30 pixels)

THEN merge two edge segments into a new segment

RULE_12

IF (1) the orientation difference of two open edge segments is less than eight degrees

(2) they do not share the same region

(3) the distance between endpoints are less than a threshold 2 (15 pixels)

THEN merge two edge segments into a new segment

After a broken edge connection, we remove the edge segments which have only one region associated with them. These are spurious edge segments because an edge segment separating regions has at least two regions associated with it.

4.4.2 Junction Finding

Junction is one of the most important features in an image, since it is invariant to translation, rotation, and the perspective projection of objects in the scene. It has played a central role in object recognition in gray-level images. Most edge detectors do not find junctions very well, because the image model of an edge is a step function of intensity discontinuity of infinite direction. At junctions, this model does not work any more. Since several regions meet at a junction, junction finding is crucial in completing the segmentation.

If there are no more edge segments which can be connected, the next step is to find junctions in the image. First, an open edge segment is checked with another edge segment which shares a common region by computing their intersection point. If the intersection point is within a specified distance (20 pixels) from either of two endpoints,

a junction is formed. Here again we can relax the error bound because two edge segments are strongly supported by the region information they share.

The remaining open edge segments are then considered for possible junctions. If the intersection of an open edge segment and another edge segment is close (less than 10 pixels) to either endpoint of the edge segments, a junction is generated. In this case we require short distance between the intersection and either endpoint is because edge segments are not boundaries of the same region. When a junction is identified, the adjacent_segment slots of both edge segments are updated with the respective edge segment. The following rules find the junctions in the image.

RULE_21

IF (1) an open edge segment borders the same region with another edge segment

(2) the distance between the intersection of two edge segments and either endpoint is less than a threshold 3 (20 pixels)

THEN create or update a junction

RULE_22

IF the distance between the intersection of two edge segment and either endpoint is less than a threshold 4 (10 pixels)

THEN create or update a junction

RULE_21 is applied first because it is easy to check if two edge segments are boundaries of the same region. Since RULE_21 is applied first, the remaining open edge segments do not share the same region with another edge segment. For example RULE_21 is applied in Figure 4.10 (a), and RULE_22 is applied in Figure 4.10 (b). The junction position is updated whenever a new edge segment joins an existing junction. The position is estimated as the weighted mean of mutual intersections of the edge segments incident to the junction, where weights are chosen according to the length of the edge segments. The coordinate slot of the group of edge segments forming the junction is also updated by this value. The junctions are classified as Arrow, Y, or L junctions depending on the angles between the edge segments.

4.4.3 T Junction

We explained in section 4.3 that a T junction plays a central role for the decision of visibility of two surfaces forming the T. The T junction provides clues on how to separate different objects in the image. Edge detection invariably misses the T junction for the same reason it has difficulty in finding other junction types.

From the remaining open edge segments, which already are assigned a convex or a concave label, we have to check if the open end can be extended to a nearby edge segment. If the distance between the intersection of the two edge segments and the end point of the open edge segment is less than 20 pixels,

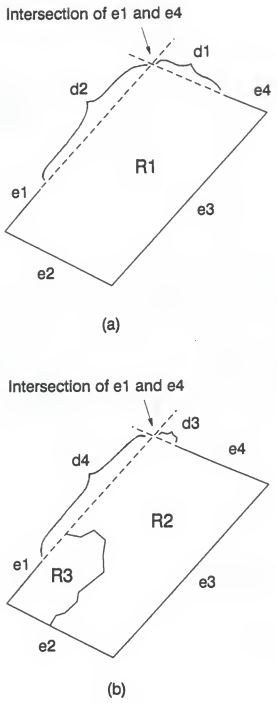


Figure 4.10 Junction finding

a T junction is created. The open edge segment becomes the shaft of the T, and the other edge segment becomes the top of the T. In Figure 4.11 edge segments e1 and e2 form two T junctions (T1, T2) with the edge segment e3. Edge segments e1 and e2 become the shafts of the T junctions and e3 becomes the top of the T. The region which contains the shaft of the T is occluded by the region containing the top of the T. Region R1 and R2 are occluded by region R3. Two edge segments forming the T junction and the location of the T are stored in the T junction table and used for the surface cluster graphs generation, which will be discussed in section 4.4. Rule_23 finds the T junctions.

RULE_23

IF (1) an open edge segment is assigned a label
(2) the distance between the intersection of two edge segments and the endpoint of the open edge segment is less than 20 pixels
THEN create a T junction

4.4.4 Region Merging and Splitting

The above procedures will produce a segmented image with some regions completely surrounded by edge segments. First we find those regions for region merging. This is similar to a polygon finding problem in a gray level image, which is usually implemented as an undirected graph search for closed

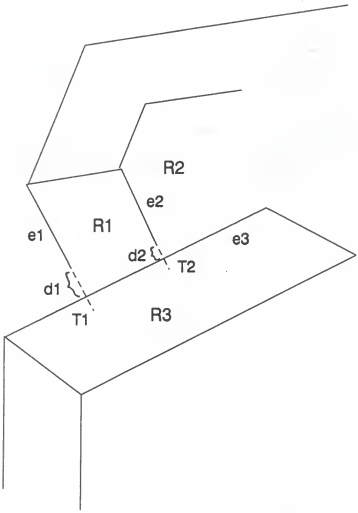


Figure 4.11 T junction finding

paths. Given junctions and line segments as a graph G(V,E), polygon finding tries to identify the shortest path from a junction i back to itself under the conditions that no line is traveled twice on the path and no line is enclosed inside the closed path. In our case, polygon finding is easy because the region information associated with the edge segments provides clues. Clockwise traversal is used in our algorithm. We start from the largest region. We pick a junction and an edge segment which borders the region such that the region is always on the right side as we traverse it. At the next junction, an edge segment which borders the same region is selected as the next segment of the polygon. This process is repeated until the traversal returns to the starting junction. If we cannot find an edge segment which borders the region, we select an edge segment which makes the largest clockwise exterior angle with the previous edge segment. For example, in Figure 4.12 the polygon corresponding to region R4 is easily found because every edge segment of the polygon (e5, e8, e9, and e10) is associated with region R4. The polygon consisting of regions R1 and R2 is found by starting with one of the junctions J1, J2, J5, or J6. Since at junction J3 the next edge segment e3 or e7 is not a region boundary of R1, the edge segment e3 is selected as the next edge segment because it makes larger clockwise exterior angle. Once a string of edge segments enclosing the region is found, bounding_edge slot of the region is filled with the edge segments string.

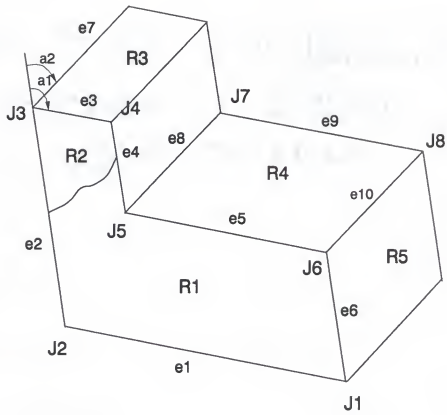


Figure 4.12 Polygon finding

Polygon finding stops when a certain number of polygons (20) are extracted.

Now we have to check whether each region can be merged with an adjacent region to form a larger region or a large region should be split into smaller regions. Since our surface classification is based on local surface property, our region homogeneity criterion is whether a region can be fitted with a first-order (planar) or a fourth-order (biquartic surface) polynomial surface function within an error bound. The general rule for trying to merge two regions is given by the following rule.

RULE_30

IF (1) a large region (size is greater than 500 pixels) is completely surrounded by edge segments
(2) the large region has an adjacent region which is inside the area enclosed by the same edge segments

THEN try a merge

A region is inside an area enclosed by edge segments if it has a non-edge boundary with the large region and edge boundaries with the other regions. This rule is actually implementing the divide and conquer principle. If a large region is surrounded by the edge segments with strong contrasts, it usually implies that the whole region is one

entity whether it be an object or a surface patch of the same curvature property. Focus of attention is given to the large regions and region merging is attempted for those regions inside the larger regions. We can achieve substantial computational efficiency by first examining an area which is likely to be one entity in terms of surface properties.

Sometimes it is very difficult for the segmentation process to correctly segment the area of a convex ellipsoid where the surface normal is parallel to viewing direction. Since surface normals around this area change very slowly, the area is usually classified as a plane area. Similarly the center of a cylindrical area where surface normals change slowly is sometimes classified as a plane area. Another candidate for merging is a small region adjacent to a large region. The following rules try to merge these regions. Merge is not allowed if the region size is greater than 900 pixels or two regions have an edge segment as their border.

RULE_31

IF a convex ellipsoid contains a plane region

THEN try a merge

RULE_32

IF a cylindrical region contains a plane region

THEN try a merge

RULE_33

IF (1) a region size is less than 200 pixels

(2) an adjacent region size is greater than 1000 pixels

THEN try a merge

Merge is allowed if the error of the polynomial fit of the merged region is within an error bound. We use two types of polynomials, plane and biquartic. For a planar surface type, we use the planar fit; for other surface types we use the biquartic surface fit. For both cases, we define the fitting error as the average angle error between the computed and measured normals from the samples of the given region:

$$\text{Error} = \sum_{i \in R} \frac{\cos^{-1}(N_i \cdot \bar{N}_i)}{M}$$

where N_i is the measured normal from the surface orientation map, $\bar{N}_i$ is the normal computed from the planar or biquartic fit, R is a region to be fitted, and M is the number of points in R involved in the fitting (200 samples).

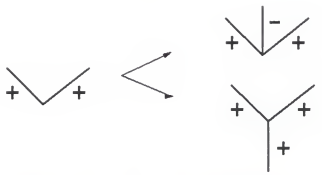
We can employ line labeling techniques to predict possible missing edges. As we discussed in section 4.3, a concave edge in surface orientation map may be a real concave edge or an occluding boundary. Since convex edges do not have ambiguities, a junction which have two or three convex edges is first identified. We then propagate the edge label constraints to neighboring junctions. Inconsistent junctions are noted for region splitting. An inconsistent junction is

a junction where line labels incident to the junction are not legal labels in the junction dictionary. For example, in Figure 4.13 (a) an L junction with two convex edges can not occur. By referring to junction dictionary we can deduce that either the concave edge of the arrow junction is missing or the convex edge of the Y junction is missing. The remaining regions are also fitted by using appropriate polynomials. If the surface fitting error is out of bound, a region should be split. A good place to split the region is at an inconsistent junction. For example, if the fitting error of region R3 in Figure 4.13 (b) is above an error tolerance, region R3 can be split at the inconsistent junction J2. Another place to look for is an open edge segment inside the region. Edge contours of weak contrasts are stored in background data base during the edge detection. If a weak edge contour is inside the region, splitting the region is tried along the edge contour. If a planar fit is successful, the surface normal slot of the region is updated by the normal of the fitted plane.

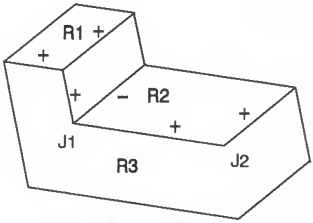
4.5 Surface Cluster Graph

4.5.1 Building the Surface Cluster Graphs

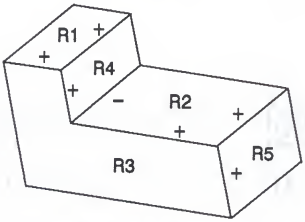
At the end of the integrated segmentation, we obtained a high level symbolic representation of a scene in terms of a region adjacency graph (RAG) whose nodes represent the segmented surface patches, and whose arcs express adjacency and geometric relationship between surface patches. Through



(a) example of an inconsistent junction



(b) J2 is an inconsitent junction



(c) R3 in (b) is split into two regions

Figure 4.13 An inconsistent junction

the integration of surface-based and edge-based segmentation, we found explicit edge types. If multiple occluded objects are in the scene, they must be separated from each other for recognition. The edge type provides clues to separate different objects. The surface patches are grouped into another graph structure called a surface cluster graph (SCG), where all the nodes in the graph belong to a single object. A surface cluster graph is a subgraph of a region adjacency graph such that every node in the SCG is connected to another node by at least one convex edge. A SCG corresponds to a distinct object in the image. Building surface cluster graphs from a region adjacency graph is carried out by reasoning, using the type of connections between adjacent surface patches. In a segmented image, the types of adjacency between two surface patches convey information regarding whether or not these two patches belong to the same object. Let S_i and S_j be two surface patches adjacent to each other. The following rules hold.

1. S_i occludes S_j if their common boundary is a limb or part of a T junction.
2. S_i and S_j are connected if their common boundary is a convex edge.
3. S_i and S_j may or may not be connected if their common boundary is a concave edge.

In surface orientation map interpretation we do not know whether orientation discontinuity in the image is from a depth

discontinuity (jump edge) or from a real convex or concave edge. Since this decision can only be made with the knowledge of other surface patch relationships, we initially interpret concave edges in the image as the place where two separate surfaces belonging to two different objects meet until other evidences are encountered.

We created a node for each surface patch. This node contains the geometrical information about the surface patch such as its surface type, orientation, and location. The labeled image regions in the RAG are ordered by the region size. A large region size in the image implies a reliable entity in the image and importance in the recognition strategy. A SCG starts with the largest region called a primary region in the image, then extend SCG building into adjacent regions. First, neighboring regions which share convex edges with the primary region are directly linked with the primary region, denoting that they belong to the same object. These new regions become the leaves of the expanding graph. Next, new regions which share convex edges with the leaves of the graph are added and become leaves of the graph. This process of adding new regions which share at least one convex edge with the existing regions is repeated until all the leaf nodes do not have any more regions sharing convex edges. This completes one graph which can be associated with one object in the model data base. The largest region which remains in the image becomes the new root node in building

another surface cluster graph. This root node expands the remaining nodes in the same way. This process continues until all the nodes in the image belong to one of the surface cluster graphs. We obtained a partition of the original region adjacency graph into a set of subgraphs (SCG's) with no links between them, each subgraph representing one object. Note that each subgraph may have a smaller number of nodes than an object actually has in the image, because we stop expanding a subgraph when we encounter a concave edge.

4.5.2 Computation of Surface Properties

The final segmentation of a surface orientation map results in two dimensional regions in the image. However, in surface based segmentation, each region corresponds to a surface patch of an object and has distinct 3-dimensional surface characteristics such as surface normal, Gaussian curvature, and mean curvature derived from the surface orientation map. Since an image is a 2-D projection of 3-D objects, the geometric features such as area and perimeter of a 3-D surface in the image varies depending on the viewpoint. For example, an area of a planar region in the image is the foreshortened area of the actual area depending on the angle between the surface normal and the viewpoint. The same holds for the perimeter of the same region. Thus the compact ratio computed from a 2-D image area and perimeter does not reflect the actual compact ratio. Actual values must be recovered

using the surface normal. For an unoccluded planar surface we can recover the actual area and perimeter using the surface normal of the plane. We assume that distances to objects are large compared to the focal length of the camera, which holds for most of the situations.

Suppose a planar region R1 consisting of m line segments in an image is a projection of a planar surface S1 (see Figure 4.14). Each line segment in the image is a 2-D projection of the corresponding 3-D edge whose z component is lost. If $\mathbf{L1} = x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}$ and $\mathbf{L2} = x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}$ are vector representation of two adjacent 3-D edges of S1 in the image coordinate system, their corresponding lines (l_1, l_2) in the image will be

$$\begin{aligned} l_1 &= x_1\mathbf{i} + y_1\mathbf{j} \\ l_2 &= x_2\mathbf{i} + y_2\mathbf{j} \end{aligned} \tag{4.1}$$

Since the surface normal **N** of S1 is perpendicular to any line on S1, the following equation holds

$$\begin{aligned} \mathbf{N} \cdot \mathbf{L}_1 &= 0 \\ \mathbf{N} \cdot \mathbf{L}_2 &= 0 \end{aligned} \tag{4.2}$$

By recovering the z components of 3-D edges we can compute the 3-D perimeter and angles of the 3-D planar surface. From (4.2) we compute the z components of $\mathbf{L}_1$ and $\mathbf{L}_2$

$$\begin{aligned} z_1 &= (x_1 \cdot N_x + y_1 \cdot N_y)/N_z \\ z_2 &= (x_2 \cdot N_x + y_2 \cdot N_y)/N_z \end{aligned} \tag{4.3}$$

where N_x, N_y, N_z are the three components of surface normal **N**. The angle between two adjacent 3-D edges is

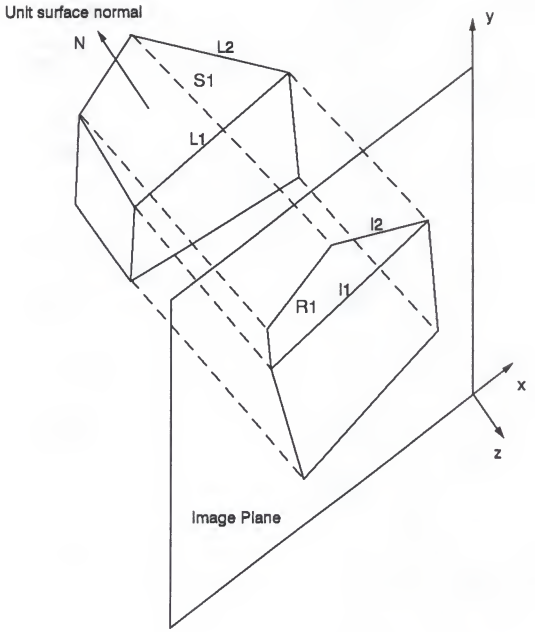


Figure 4.14 A surface and its projection on an image

$$\angle L_1L_2 = 180^\circ - \cos^{-1}(L_1 \cdot L_2) \qquad (4.4)$$

The area of an image region R1 is a foreshortened 3-D area of S1. They are related by the cosine of the angle θ of the surface normal **N** of S1 and the viewing vector (z axis).

$$\text{Area of R1} = (\text{Area of S1}) \cdot \cos\theta \qquad (4.5)$$
$$\cos\theta = (0, 0, 1) \cdot (N_x, N_y, N_z) = N_z$$
$$\text{Area of S1} = \text{Area of R1} / N_z$$

Of course, the length of the line and area of the plane are not the absolute length and area of the original surface, but if we have the approximate distance information to objects before hand (as on a factory assembly line), we can estimate the absolute area by multiplying the above area by a scale factor. Similarly, the other useful 2-D features such as moments and compact ratios can also be computed. These computed surface features are put into the appropriate slots in the surface representation and later used for recognition.

4.6 Related Research

A needle map has been used for object recognition by several researchers [Horn 84, Horn and Ikeu 84]. A needle map is generated by a photometric stereo procedure, which determines the surface normal vector at each pixel, using lookup table indexed by the three intensity values of the

three images taken from different illumination directions. Segmentation of the needle map is accomplished by the photometric procedure, which distinguishes regions where surface normals are computable from those regions where surface normals are not computable. A surface normal is not computable when triplets of brightness values are impossible combinations due to shadow or mutual illumination. This information is also stored in lookup table to be used for segmentation. Each surface normal vector is then translated to the center of a unit Gaussian sphere. By associating a unit mass with each surface normal vector, mass distribution of surface normals of an object is obtained. This mass distribution of surface normals is called the Extended Gaussian Image (EGI). The EGI is one way of globally representing an object. A 3-D object is modeled using a set of EGIs, one for each possible viewing direction on a uniformly sampled viewing sphere (multiview representation). Matching is performed by comparing an observed EGI from an image with each model EGI, which is computationally expensive due to multiview representation. The EGI approach do not make explicit use of surface and adjacency information, which are vital for the object recognition of a large number of model objects. This approach is applicable only for convex objects without occlusion, and it can not distinguish certain shapes [Besl and Jain 85].

Besl and Jain [Besl and Jain 88] presented a surface-based segmentation technique for range images. They used view independent surface characterization based on the Gaussian and mean curvature signs. The signs of Gaussian and mean surface curvatures are used to provide initial coarse image segmentation into eight fundamental surface types: peak, pit, ridge, valley, saddle ridge, saddle valley, flat, and minimal (see Table 3.2 in Chapter 3). Then the segmentation is refined by a fitting and growing algorithm. For each connected region of a given surface type, a seed region is first found by iteratively shrinking the connected region until the area of the largest region is smaller than a threshold. Then iterative region growing is applied to fit the original image data with bivariate polynomials up to the fourth order. Region growing stops when the fitting error exceeds a threshold or further growing does not include enough pixels. Finally, surface regions that join smoothly at their shared boundaries are merged together to create the final surface region description. Since they use the signs of Gaussian and mean curvatures only, adjacent distinct surfaces may be segmented into the same surface type. They avoid this problem by seed region extraction and then region growing which is computationally expensive.

Several integration approaches are tried for gray level image segmentation. Levine and Nazif [Levi and Nazi 84] used an expert system approach to combine edge detection and region

based segmentation of a gray level image. Pavlidis and Liow [Pavl and Liow 90] first segmented a gray level image using split and merge algorithm and then region boundaries were eliminated or modified on the basis of edge contrast and smoothness. Since these approaches used gray level images only, the resulting segmentation still contain errors due to noise and can not provide a rich description on 3-D surfaces.

Compared to the EGI approach, our integrated segmentation provides rich description on the distinct surfaces and their well defined boundaries, which are essential information for object recognition even in the presence of occlusion. In comparison with the Besl and Jain's approach which uses only the signs of Gaussian and mean curvatures, the use of high curvature regions in our approach provide reliable surface segmentation at an early stage. If we use signs of Gaussian and mean curvatures only, two distinct neighboring surfaces (e.g., two adjacent planes) may be segmented into a single region. Also, by integrating the edge output with the surface segmentation, we removed computationally expensive operation of seed region extraction and the subsequent region growing.

4.7 Experimental Results

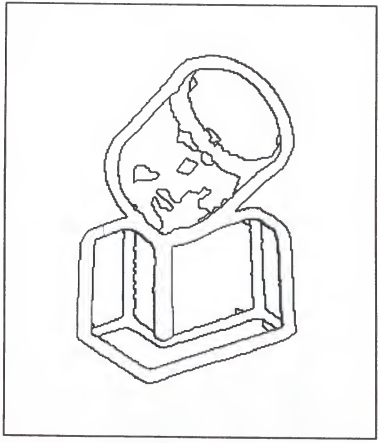
The integrated segmentation paradigm developed in this chapter is applied to several images. Figure 4.15 (a) is the refined surface-based segmentation result of a stair and cylinder image using extended Gaussian(K) and mean(H)

curvatures discussed in section 3.4. Figure 4.19 (a) is the refined surface-based segmentation of a stair and trough image. After connected component labeling, a region adjacency graph is constructed for region operations. Small regions are merged with adjacent non-high curvature regions. The result of small region merging is shown in Figure 4.15 (b) and Figure 4.19 (b). Surface-based segmentation and edge-based segmentation are then integrated to obtain a more reliable and coherent segmentation. Edge type (convex or concave) is decided by examining where an edge segment lay in surface-based segmentation. Figures 4.16 (a) and 4.20 (a) show the superimposition of labeled edge segments with the surface-based segmentation. Note that edge segments which do not lay inside high curvature regions are considered as spurious edges. Non-high curvature regions are then expanded into high curvature regions until encountering either an edge segment pixel or another region pixel. Figures 4.16 (b) and 4.20 (b) show the segmentation result after high curvature region removal. Compare these with the edge detection outputs of Figures 2.7 (a) and 2.12 (a). Note that all the region boundaries are closed. More importantly, the surface type of each region is known. The edge and region association is then carried out by tracing the region boundaries. Broken edge segments are connected using the region information associated with the edge segments. Edges which do not have regions associated with them are spurious edges. These edges are

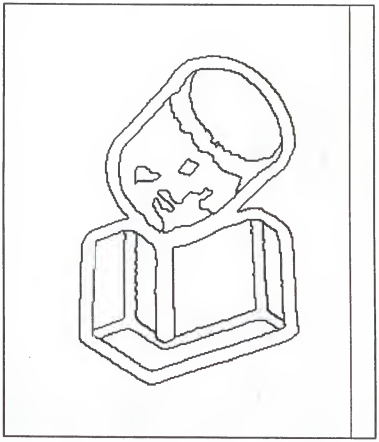
inside regions and are removed. Figures 4.17 (a) and 4.21 (a) show the result of broken edges connection and of spurious edges removal. Junctions are then found from open edge segments using the region information associated with the edge segments. Figures 4.17 (b) and 4.21 (b) are the result of junction finding. By using the region information associated with the edge segments, we eliminated accidental connection of broken edge segments and found junctions reliably and fast. Open edge segments associated with the junctions are extended to the junctions. Remaining open edge segments are extended to other edge segments using the line equations or the conic equations associated with the edge segments. These equations are derived during edge contour segmentation by the line or conic fitter. Figures 4.18 (a) and 4.22 (a) are the result of open edge segments extension to junctions and other edge segments. Finally, region merging rules are applied to merge regions inside a large region bounded by edge segments. Figures 4.18 (b) and 4.22 (b) are the final segmentation result. Figure 4.23 is the initial and final segmentation results of a bulb image.

From the final segmentation we build surface cluster graphs of distinct objects in the image. The final segmentation of a stair and cylinder image is shown at the top of Figure 4.24. Note that, through the integrated segmentation, a node in the RAG has area and surface type information about a surface patch, and an arc has edge type

and adjacent region information. The largest region identified is face F2 of the cylinder. From the adjacent regions we find that only F1 has a convex edge with F2. Since there are no more convexly adjacent regions, the first surface cluster graph consisting of F2 and F1 is built. From the remaining regions we find that F5 is the largest region. Another SCG building starts with F5. F6 and F7 are convexly adjacent to F5, so they are added to F5. Note that the adjacent F4 is not added to F5. Since F3 and F4 are convexly adjacent to F7 they are added to the SCG. No more regions are convexly adjacent to the existing regions of SCG and the second SCG is built, shown as the surface cluster graph of the stair. The RAG of a stair and trough image is shown in Figure 4.25. F1 is the largest region and, it is extended to convexly adjacent regions F2, F3, F4, F5, and F6, forming the first SCG. The second SCG is similarly built starting from F8. Solid arcs denote convex edges and dashed arcs denote concave edges. Each surface cluster graph represents a different object in the scene. The recognition process starts with these surface cluster graphs to recognize and verify the objects in the scene.

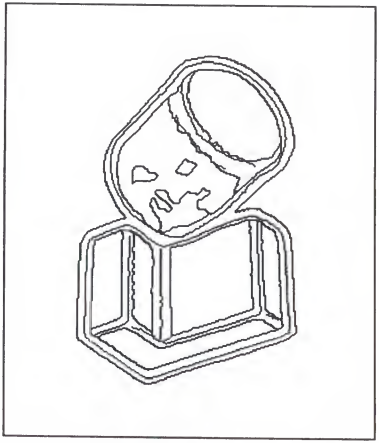


(a) Segmentation after shrinking and expansion

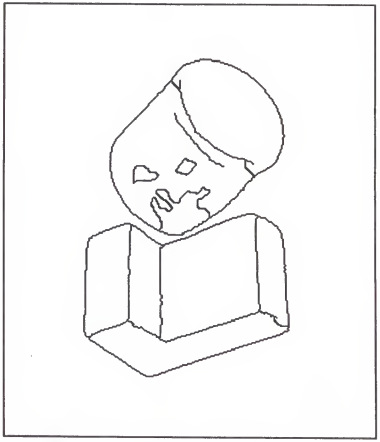


(b) Small region merging

Figure 4.15 Small region merging

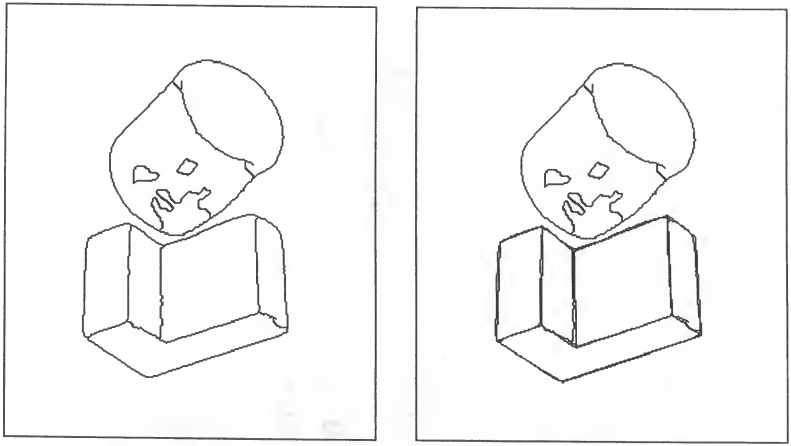


(a) Registration of edge segments with the surface based segmentation



(b) High curvature region removal

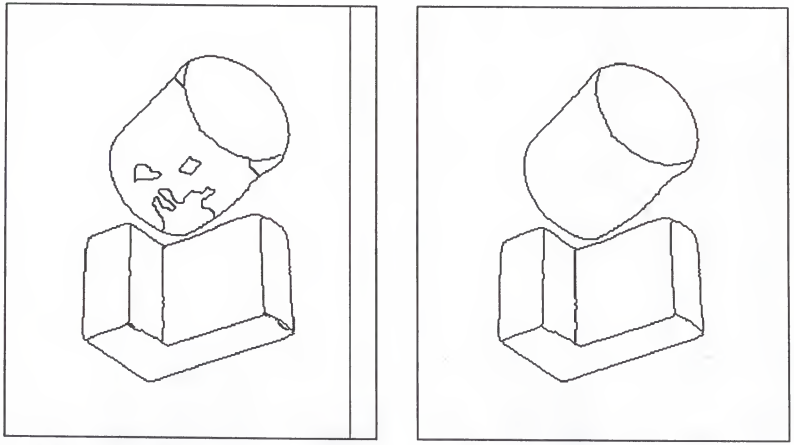
Figure 4.16 Superimposition of edge segments with surface segmentation



(a) Spurious edge removal

(b) Connected junctions

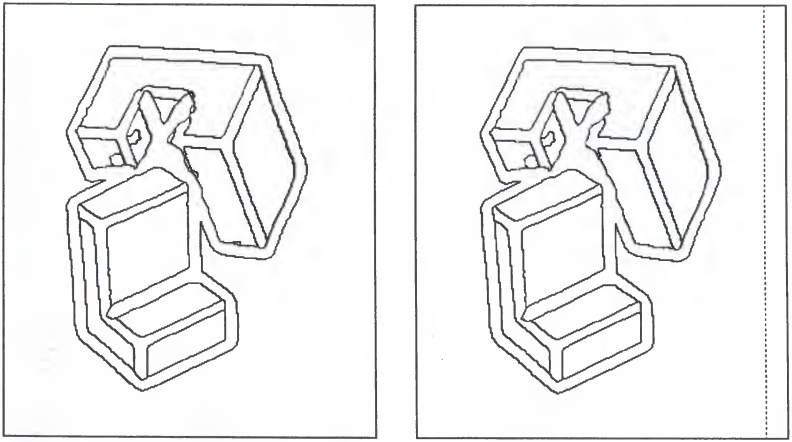
Figure 4.17 Sprurious edge removal and junction finding



(a) open edge segments extension

(b) Final segmentation

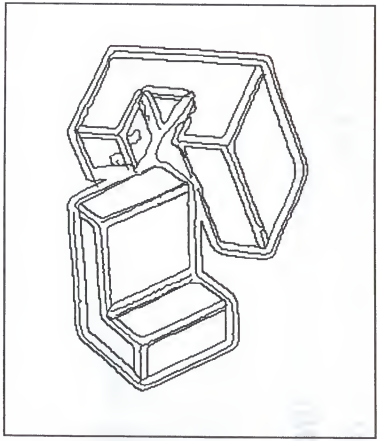
Figure 4.18 Extension of open edge segments and final segmentation



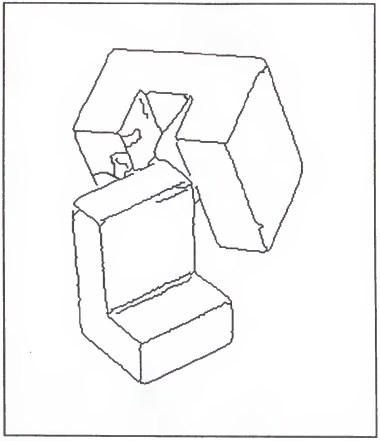
(a) Segmentation after shrinking and expansion

(b) Small region merging

Figure 4.19 Small region merging

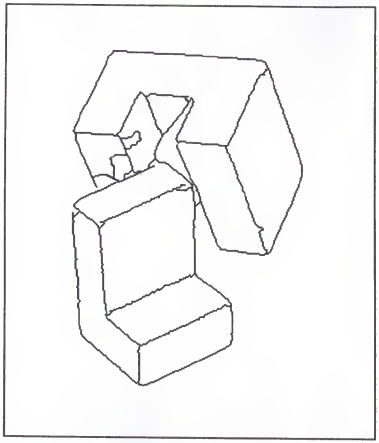


(a) Registration of edge segments with the surface based segmentation

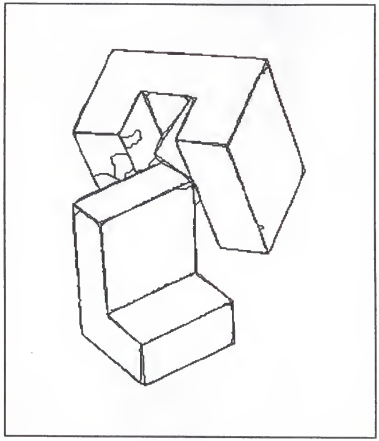


(b) High curvature region removal

Figure 4.20 Superimposition of edge segments with surface segmentation

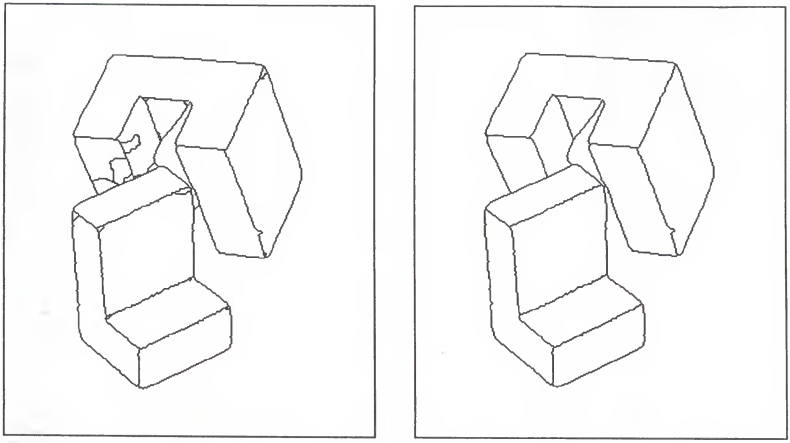


(a) Broken line connection and spurious edge removal



(b) connected junctions

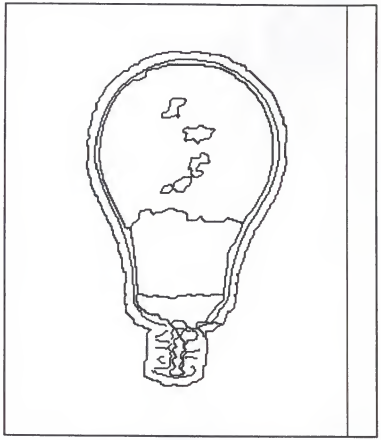
Figure 4.21 Sprurious edge removal and junction finding



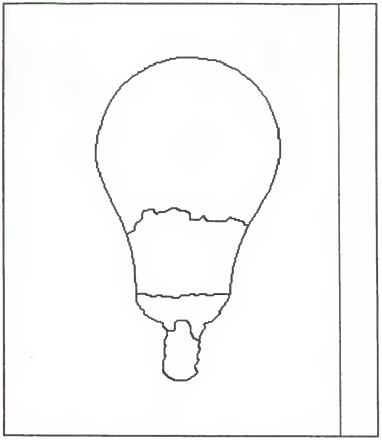
(a) open edge segments extension

(b) Final segmentation

Figure 4.22 Extension of open edge segments and final segmentation

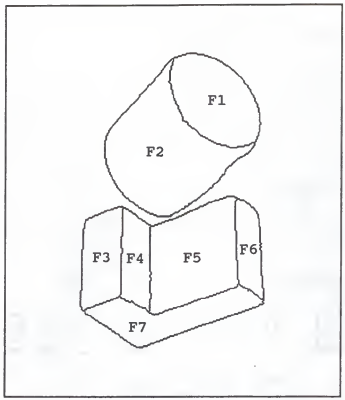


(a) Edges superimposed on refined segmentation

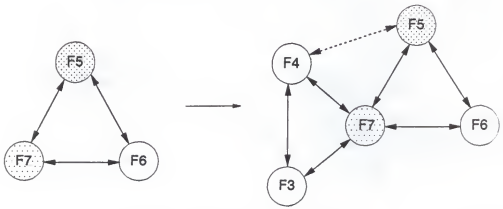


(b) Final segmentation

Figure 4.23 Integrated segmentation of a bulb image

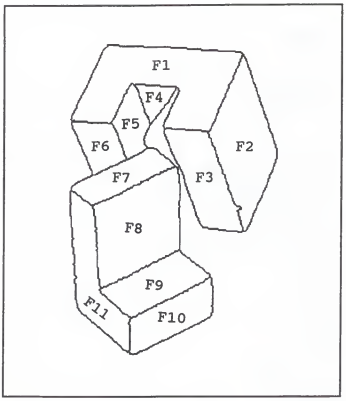


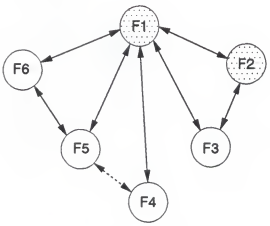
Surface Cluster Graph of the Cylinder



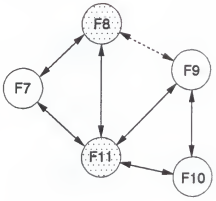
Surface Cluster Graph of the Stair

Figure 4.24 Surface cluster graphs of a cylinder and a stair





Surface Cluster Graph of the Trough



Surface Cluster Graph of the Stair

Figure 4.25 Surface cluster graphs of a trough and a stair

CHAPTER 5
RECOGNITION

Recognition implies that a correspondence has been found between the elements of an image and a representation of an object in the world. Recognition can be achieved through correspondences between many kinds of predicted and measured features such as shape, color, texture, connectivity, context, motion, shading, etc. The single most important of these properties is probably the shape property. In 3-D object recognition, features extracted from a 2-D image (in the viewer-centered coordinate system) do not immediately correspond to object models (in the object-centered coordinate system). 3-D to 2-D, or 2-D to 3-D transformations must be performed before the observed features in the image can be matched with the model. But this transformation can not be determined until the recognition is complete, because exact correspondences between image features and model features are used (which means that the object is recognized) to compute the transformation. The computation of the transformation is one of the central issues in recognition, because verification of recognition can only be achieved by converting the 3-D object model to a 2-D image through the transformation, and then features can be compared. In addition to verifying the

recognition, the transformation provides the location and attitude information of the object with respect to a viewer centered coordinate system, which are necessary for sophisticated tasks such as assembly and navigation.

Without explicit 3-D information, the recognition is usually achieved by the hypothesize-and-verify paradigm, where a viewing transformation is computed based on a few hypothesized matches. Then, other image features are used for verification. The number of hypotheses grows exponentially as the number of objects in the model data base increases. The situation gets further complicated if multiple objects in the scene are occluded by each other.

There are several important interrelated issues which should be addressed in the design of a recognition system [Chin and Dyer 86]. Resolving these issues determines the speed, power, and flexibility of the recognition system.

1.Sensor model and feature detection

Various kinds of sensors are now used in object recognition systems, including optical sensors, range finders, and tactile sensors. These sensors transform object features into image features. Sensor detectability specifies what kinds of features can be detected and under what condition they are detected. Sensor reliability is a measure of uncertainty in the detected features. For a particular sensor we have to answer the question "What features should be extracted from an image

in order to adequately describe its physical properties and their spatial relationships in a scene?"

2. Model representation

What constitutes an adequate representation of these features and their relationship for characterizing a semantically meaningful class of objects? That is, in what form should these features be combined into object models such that this description is appropriate for recognizing all objects in the given class ?

3. Matching

How should the matching or correspondence or be done between image features and object models in order to recognize objects in a complex scene ?

Various recognition strategies have been developed depending on the image sensor used, model representation, matching strategy, and system requirements.

5.1 Review of 3-D Object Recognition

Object representation plays a central role in a recognition system design. The recognition system can be broadly classified into three classes on the basis of their spatial description: 2-D, 2.5D, and 3-D representations.

2-D spatial descriptions are viewer-centered representations in image space. Each distinct view is represented using, for the most part, 2-D shape features,

which are derived from a gray-scale or from the binary image of a prototype object. This class of representation is appropriate when the viewpoint is fixed and only a small number of stable object positions are possible. The 2-D representations are further subdivided into three classes according to their method of object modeling. They are (1) the global feature method, (2) the structural feature method, and (3) the relational graph method. Global features include perimeter, centroid, distance of contour points from centroid, curvature, area, moments of inertia, and Fourier descriptor. Examples of local features include line segment, arc segment with constant curvature, defining pieces of object boundary, and holes. Relational features include a variety of distance and relative orientation measurements, interrelating the substructures and regions of an object. Models based on the geometric properties of an object's visible surfaces or silhouette are commonly used because they describe objects in terms of their constituent shape features. 2-D models have the advantage that they can be automatically constructed during the training phase from a set of prototype objects, one from each possible viewpoint. Their disadvantage is that they do not make the full 3-D description of the object explicit--their completeness depends on the complexity of the object and the number and positions of viewpoints used.

2.5-D representations have attributes of both 2-D and 3-D representations. These spatial descriptions are viewer-

centered representations of surfaces, but depend on local surface properties of the object in each view, for example, range (depth) and surface orientation. They have 3-D properties but those properties hold only for one particular view.

3-D spatial descriptors define the exact representations in object space using an object-centered coordinate system. 3-D representations are view-point independent volumetric representations permitting computation at arbitrary view-points. Computing complete 3-D description is very difficult because we can only see the front half of the objects. The other half must be interpolated, or we may need several different views to obtain the complete volume description.

Now, several recognition systems which have close relationships with our research are reviewed. Tou and Fan [Tou and Fan 88] developed an elegant method, called geo-graphic code, for the representation and recognition of polyhedral objects. The geo-graphic code is a multiview representation, where the encoding captures the topological and geometrical properties of an object in a particular view. The geo-graphic code consists of six entities: (1) the number of visible vertices, (2) the number of visible surfaces, (3) the vertex-surface description (VSD), where an object is encoded in terms of the number of surfaces and of the vertex sequence of each surface, (4) the vertex-surface vector (VSV), which represents the number and order of surfaces within the

VSD, and the vertex number of each surface, (5) the vertex-type-description (VTD), which is generated from VSD by replacing the vertex label by vertex type, and (6) the vertex-link code (VLC) where each edge linking two vertices are represented by 64 direction code. This representation is very good for a single polyhedral object recognition but difficult to use when multiple curved objects are occluded in the scene. The main usage is only for recognition, and the orientation and location of the recognized object are not available.

Tou and Huang [Tou and Huan 86] proposed orthographic projection (OP) views as 3-D object recognition. By making use of the regularity constraints and the property of the gradient space, the orientation of each surface is derived from the pictorial drawing of an object in an image. The footprints of the 3-D object, that is to say, the top view, the front view, and the side view, are then generated by integrating the surfaces projected on each view by orthographic views transformation. They are unique representations of a 3-D object, which can be derived from CAD/CAM data base or by training during model generation. But the OP views generated from a pictorial drawing may not be unique for an arbitrary viewpoint and may require several different views of the same object. This representation is not suitable when multiple objects are occluded in the image. In this dissertation, we extend this concept and use the surface and edge features which are reliably extracted from

the integration of edge detection and surface segmentation of the surface orientation map.

A relational-feature graph based recognition system was developed by Oshima and Shirai [Oshi and Shir 83], where nodes in the graph represent planar or smoothly curved surfaces extracted from a range map, and arcs represent relations between adjacent surfaces. The basic surface types include planar, cylinder, ellipsoid, hyperboloid, cone, paraboloid, and others. For each pair of adjacent regions, the type of intersection (convex, concave, mixed, or no intersection), angle between regions, and the relative positions of the centroids are stored. If objects may be viewed from multiple viewing positions, then a separate relational graph must be constructed for each view, and these models must be treated independently. Matching is performed by comparing an observed relational graph of surface descriptions with a set of graphs of each viewpoint of each object modeled. A kernel representing regions with a high confidence of being found is compared with all model graphs to select candidate models. Finally, the system performs a depth-first search which attempts to determine the correspondence between each remaining region in the current candidate model and the regions extracted from the scene.

Object recognition based on special purpose automatic programming was presented by Goad [Goad 83]. Individual object descriptions are compiled into programs for which the only

task is recognizing one object from any view. Time consuming shape analysis is performed off-line prior to the recognition phase to reduce recognition time. He uses a multiple-view object feature model that incorporates 218 different 3-D views of each object. The features are line segments stored as a pair of endpoints and a 218 bit string. The bit string describes the visibility of the feature in each of 218 discrete views. Edges for objects are ordered by their expected utility for matching purposes.

Tropf and Walter [Trop and Walt 83] discuss an augmented transition network (ATN) model for a single image recognition of randomly oriented 3-D solid objects with known geometry. The ATN model is used to control an analysis-by-synthesis search procedure based on hypothesis generation and verification. Features used are point primitives such as corners or vertices. First a parallel projection of the points is created from an arbitrary view. A point is picked from the projected data and it is hypothesized that it is a point P1 on a particular known object from object library. That object must have a second point P2, which is a distance R from P1. In 3-D, P2 must lie on the surface of a sphere of radius R; in the 2-D projected image, P2 must lie within a circle of radius R. A second point is chosen within that circle, if one exists. Otherwise, another second point of the object is tried. If none of those work, the next object is tried. Now it is assumed that the axis P1-P2 in 3-D space is

known. Next a third point P3 on the object not lying on P1-P2 is considered. It must lie on a 3-D circle surrounding the P1-P2 axis and must therefore lie on a known ellipse in the projected image. Now a point is picked in the image closest to the ellipse that will fix the object in space. Object verification is the final step in the process. The ATN approach has the potential to cope with heavily distorted and noisy image data.

The polyhedral object recognition and localization problem as an interpretation tree (IT) search was treated by Grimson and Lozano-Perez [Grim and Loza 84]. Assuming sparse positions of points on a surface of one of the m model objects, an IT is built where the root node is the object and the first n_j descendants are the n_j features of an object. The next level of the interpretation tree is for different features, and IT building continues until all of the features are accounted for. Interpretation consists of the pairings of image features and model features, that is to say, a depth-first search through the IT. Interpretations inconsistent with local constraints are discarded. Interpretation continues until a reasonable number of consistent pairings have been made. Final verification is carried out by a transformation test.

Most of these methods are general and robust because only very primitive features are used, which can be reliably and redundantly detected such as lines, corners, and surface

normals. But these methods pay a price in the matching phase where an almost blind search is used. The reason is that structuring low level features into higher level entities is still very difficult and unreliable. Our integration of the two channel output is an attempt to structure low level features reliably. This integration is rewarded in the recognition phase that provides the possible object candidates which may explain the observed image features.

5.2 Geometric Model

To recognize objects, one must have an internal representation of an object suitable for matching its features to image descriptions. The 3-D object representation plays an important role in many scientific and engineering fields. The difficulty of three-dimensional object recognition lies in the fact that objects are three-dimensional in nature and that the image is two-dimensional. The appearance of a 2-D projection of an object changes with the viewpoint. Ideally, an extracted three dimensional volumetric description from the image could be directly compared to the three dimensional model. This is a very difficult task. The model representation is intricately related to the recognition strategy, depending on whether occlusion is allowed or not. The objects we are dealing with in this research are solid opaque objects.

Solid representation is essential in computer graphics, computer vision and other applications. Three general classes of representations [Requ 80] have been studied, that is to say, boundary or surface representation, sweep representation (generalized cylinders), and volumetric representation (constructive solid geometry). The representations have different computational properties and efficiency, which dictates their use in certain applications. Volumetric models represent the solid components of an object in relation to each other or to the whole object. Examples include space filling models that represent objects by denoting the portions of space in which the object is located, and constructive solid geometry (CSG) models, that start from geometric primitives such as cubes, cylinders or half-spaces and then form more complex objects by union, intersection, and difference operations on these primitives. With these, the three dimensional character is directly accessible, but appearance is hard to deduce without the addition of surface shape.

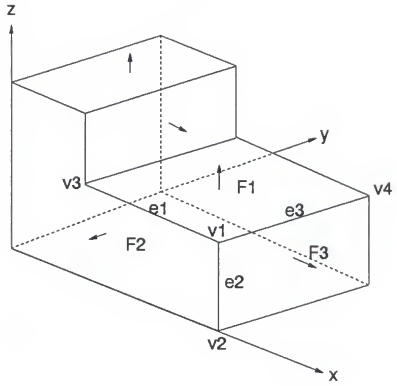
Another widely used volumetric model for computer vision is the generalized cylinder, where an object is specified by a cross-sectional shape, an axis along which to sweep the cross section, and a sweeping rule describing how the shape and orientation of the cross section vary along the axis. A criticism of the generalized cylinder/cone representation concerns its choice as a primitive element. Many natural and

man-made objects do not have vaguely cylindrical components.

Since we are dealing with verification vision, a more explicit geometric representation is needed, which easily characterizes surface properties. Boundary representation is a natural choice. We use extended CAD type boundary representation where the representation is augmented by surface feature properties for use in recognition. In boundary representation, a solid is segmented by a finite number of faces, and each face by its bounding edges and vertices. We limit ourselves only to objects which are formed by planes and quadric surfaces. Geometric models explicitly represent the shape and structure of an object.

Figure 5.1 shows the object representation scheme used in our research. Each object consists of three different levels of description: surface level, edge level, and vertex level. The surface list contains a linked list of pointers to each surface comprising the object. Each surface has pointers to the bounding edges of the surface, and each edge has pointers to the vertices at the end of the edge. The edges and vertices in turn have pointers to faces and edges, too.

Figure 5.2 shows a frame representation of face, edge, and vertex information. In addition to the bounding edge list and vertex list, the face frame contains various property slots for recognition purpose. The surface type slot represents symbolic knowledge which qualitatively describes the surface characteristic. Surface type includes plane,



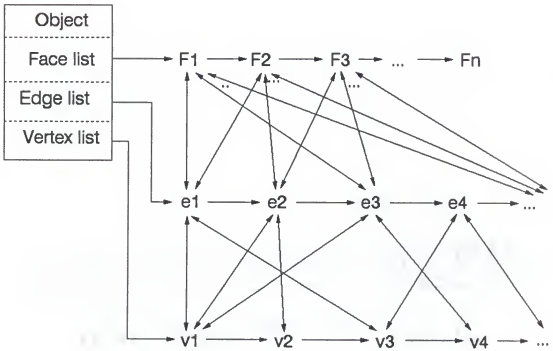


Figure 5.1 Object representation

```
(Face_Name      (Face_Type           ( Plane Cylinder,...))
                (Bounding_Edges      ( e1 e2 e3 ...))
                (Vertex_list         ( v1 v2  ...))
                (Surface_Normal      ( N1 N2 N3 ))
                (Number_of_Sides     ( 7 ))
                (Compact_Ratio       ( 1.8))
                (Area                ( 330 ))
                (Elongation_index    ( 5 ))
                (Angle_List          ( ∠e1e2  ∠e2e3 ...)))

(Edge_Name      (Edge_Type           ( Convex Concave ..))
                (Angle               ( 90 ))
                (Associated_face     ( F1 F2 ))
                (Associated_Vertices( v1 v2 ))
                (Direction_Vector    ( U1 U2 U3 )))

(Vertex_Name    (Associated_Edge     ( e1 e2 e3 ...))
                (Associated_Face     ( F1 F2 F4...))
                (Vertex_Coord        ( x  y  z )))
```

Fig 5.2 Frame representation of face, edge, and vertex

convex cylinder, concave cylinder, convex ellipsoid, concave ellipsoid, and saddle shape. The surface normal slot contains a surface normal vector with respect to a given object centered coordinate system. For nonplanar surface types this slot points to empty value. The compact ratio slot contains the ratio of the surface perimeter squared to the surface area. It is a scale invariant feature. Since we do not generally know the absolute area and the perimeter of an observed surface, this information plays a crucial role in candidate model generation. We also have a perimeter and an area slots, which are used to compare these features with those of adjacent surfaces for a relative value which validates the adjacency relationship between the two surfaces.

The edge frame contains slots for pointers to two neighboring surfaces and for two vertices at ends. Edge types include convex linear edge, concave linear edge, convex curved edge, concave curved edge, and parabolic curve. Parabolic curves are not physical edges and cannot be detected by an edge detector, but they are surface shape discontinuities where the Gaussian curvature changes sign. A parabolic curve separates two different surface types such as bulb head and bulb stem. This curve is represented as a region boundary between a convex surface and a hyperbolic or a cylindrical surfaces. The angle slot includes the angle between two neighboring surfaces. The vertex frame contains slots for pointers to edges and surfaces of which the vertex is a

member. The redundant structure of information makes the recognition system easily access various levels of information quickly during the candidate model generation phase and the verification phases.

In addition to the explicit boundary representation of object models, we construct a surface classification network to facilitate candidate model generation. Figure 5.3 shows the surface classification network. All the surfaces of object models are sorted according to their surface type. Nonplanar surface types are further classified by ranges of principal curvature values. Plane surface type is further classified according to the number of boundary sides, and, for equal number of sides, the network further classifies the type by ranges of compact ratio. We use compact ratio because it is a scale invariant feature. Each representative contains a list of pointers to objects which have the same surface type and characteristics. Since these geometric quantities can rarely be extracted reliably, each list has an upper bound and a lower bound slot. Some objects may belong to two lists.

5.3 Geometric Transformation

Geometric transformation (camera transformation) between an object model and an image is a central problem for a robust and powerful vision system which can recognize objects in the presence of occlusion among the objects. We will study the transformations involved between a 3-D space point and its

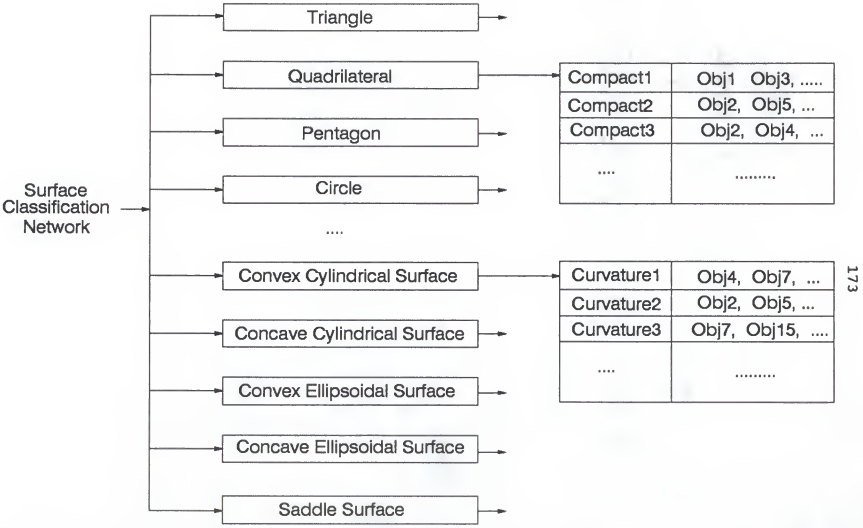


Figure 5.3 Surface classification network

projection into a 2-D image point. Point projection is the fundamental model for the transformation wrought by our eye, by cameras, or by numerous other imaging devices. To a first-order approximation, these devices act like a pinhole camera in that the image results from projecting scene points through a single point onto an image plane.

In order to treat transformations between two coordinate systems consistently and uniformly, homogeneous coordinates have been used in many fields [Paul 84]. In a homogeneous coordinate system, a fourth coordinate, or scale factor, is used in such a way that the total scale of a vector is unimportant.

A point (x,y,z) in 3-D space is represented as (wx,wy,wz,w) in homogeneous coordinates for some nonzero w. The conversion from homogeneous coordinate to 3-D space point is accomplished by dividing the first three components by the fourth component.

Geometric transformations such as translation, rotation, scaling, and perspective projection can easily be described in a homogeneous coordinate system by simply multiplying the matrix representing the corresponding geometric transformation. A complex transformation is simply the concatenation of primitive transformations. In order to perform geometric transformation in 3-D space, we first convert the 3-D point to homogeneous coordinates, manipulate all the transformations in the homogeneous coordinate system,

then recover the original 3-D space points. The transformation of the three-dimensional coordinates of a point in space to the two-dimensional coordinates of its image can be expressed compactly as a 4 by 4 homogeneous coordinate transformation matrix (actually we use 3 by 4 matrix because z component in image is a fixed distance(f) from the camera lens center). The homogeneous coordinates and the homogeneous coordinates transformation matrix are a convenient means for representing an arbitrary transformation.

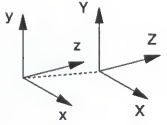
5.3.1 Elementary Transformations

In the following discussion, a vector, or point, is represented as a column vector. Left handed coordinate system is assumed unless specified otherwise.

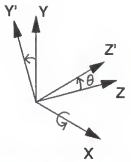
The translation transformation (Figure 5.4 (a)) which moves an origin to a point (x_t, y_t, z_t) is ((x_t,y_t,z_t) becomes an origin in new coordinate system):

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & -x_t \\ 0 & 1 & 0 & -y_t \\ 0 & 0 & 1 & -z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.1)$$

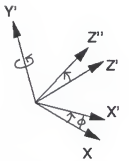
The coordinate transformation corresponding to rotation about the x axis (Figure 5.4 (b)) through an angle θ is (positive θ is defined as counterclockwise (CCW) direction) :



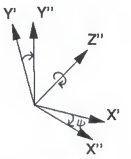
(a) Translation



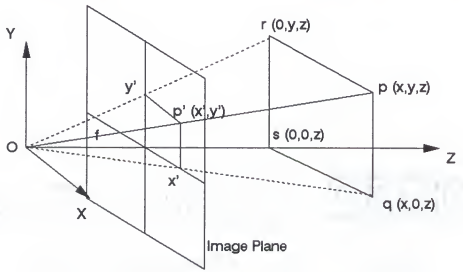
(b) Rotation about X axis



(c) Rotation about Y' axis



(d) Rotation about Z'' axis



(e) Perspective projection

Figure 5.4 Geometric transformations

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

Similarly coordinate transformation of rotations about y,z axes (Figure 5.4 (c), (d)) are defined as :

$$R_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

There is a subtle difference between point transformation and coordinate transformation. When we want to rotate point (x,y) to point (x',y') in CCW rotation, it is the same as rotating coordinate system by the same amount in CW direction. So all the above angles become negative angles when we want to view the operation as a point transformation.

In orthographic projection, a scaling transformation can be used to scale units in a real world into units in image domain.

$$S = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (5.5)$$

An arbitrary affine transformation can be composed by the combination of these transformations.

$$\mathbf{T}_{tot} = \mathbf{S}\mathbf{R}_z\mathbf{R}_y\mathbf{R}_x\mathbf{D} \qquad (5.6)$$

The perspective projection of a point **p** in a camera of a focal length f is given in Figure 5.4 (e). Using the similar triangle property we have f : x'= z : x and f : y' = z : y. Perspective projection of a point (x, y, z) is then given as

$$x' = \frac{x\,f}{z} \qquad y' = \frac{y\,f}{z} \qquad (5.7)$$

where f is focal length of camera.

5.3.2 Computation of a Viewpoint

There are seven underlying parameters in the camera transform presented above: three parameters(x_t, y_t, z_t) give the camera position **t**, three more(θ, ϕ, ψ) are sufficient to specify the rotation **R**, and a focal length f specifies a property of the camera itself. Without explicit 3-D information, the problem of finding the camera transformation between a model and the corresponding image becomes a perspective n-point problem. The perspective n-point problem tries to determine the camera transformation given hypothesized n-point correspondences between an image and a model. Since correct

correspondences can be established only after recognition, computation of the camera transformation is usually within the recognition loop, which tests all the combinations of model features and image features (hypotheses space) to verify that the transformation generates the observed image features from the model. The verified camera transformation is then used to determine the position and orientation of a camera with respect to a viewer centered coordinate system. The hypotheses space to be tested grows exponentially as the number of objects in the data base increases. The rich description of surfaces and well defined surface boundaries obtained through our integrated segmentation prunes the hypotheses space drastically using surface classification network and adjacency and geometric constraints of surfaces at an early stage.

Roberts [Robe 63] expressed this problem as a least squares problem and his solution requires seven point-to-point correspondences between a model and an image. Horaud [Hora 86] showed a constructive method for computing the viewing transformation for the special case of three lines forming a vertex.

An alignment method was presented by Huttenlocher and Ullman [Hutt and Ullm 87] to compute the view transformation by using three points correspondences. They assume an orthographic projection between an object and an image, which is equivalent to finding an affine transformation. A point in

the model and a point in the image are aligned first. Then a line in the model connecting the first point and a second point is aligned with the image line of the corresponding points restricting the transformation. Finally, the third point provides a complete transformation and a scale factor.

Fischler and Bolles [Fisc and Boll 81] present a closed-form solution for this problem and describe important results for the conditions under which multiple solutions exist for various numbers of correspondences between image and model. Fischler and Bolles noticed that for every real positive solution there is a negative solution and hence a maximum of four solutions are in fact possible. Multiple solutions may exist even for four or five matches in the general position. This means that at least six matches of points in the general position (a total of twelve constraints) are required to assure a unique solution to the six-parameter problem.

Dhome et al. [Dhom et al. 88] present another general solution for three image-line to model-edge correspondences. They solve the rotation problem first by formulating eight degree polynomial equation using the 3-line correspondences in general conditions. Depending upon the configuration of the triple of lines (coplanar, trihedral, general), there are four, six, or eight solutions. They removed the impossible solutions which lie on the negative z axis or the solutions which contain invisible lines. The 8 solutions result agrees with the result obtained previously by Fischler and Bolles.

Assuming that a camera focal length f is known by calibration, six parameters are needed to determine the orientation and location of the camera. An iterative approach tries to determine the six parameters by starting with the initial specification of **R** and **D** (for example, surface and edge correspondences between a model and image description can provide this information) and refining **R** and **D** through error measurement. Since each match between an object point and an image point constrains two degrees of freedom, only three point-to-point matches between an image and a model are needed to achieve a complete solution.

5.3.3 Iterative Solution

An explicit surface description of the surface cluster graph generated in chapter 4 provides very strong clues for the orientations and the types of objects in the scene. But we need an incremental least squares type solution to accumulate evidences from various image features in order to solve the occlusion problem in the scene. We discuss an elegant iterative solution [Lowe 86] to the camera transform problem.

Following the standard techniques used in graphics, let **t** denote the location of the camera in world coordinates, let **R** ($R_z R_y R_x$) be the rotation transform which specifies the camera orientation, and let f be the focal length of the camera. A point **p** in world coordinate is first transformed into the point (x, y, z) in camera-based coordinates.

$$(x, y, z) = \mathbf{R} (\mathbf{p} - \mathbf{t}) \tag{5.8}$$

The perspective projection of this point onto the image plane, with image coordinates (x', y') is given by (see Figure 5.4(e))

$$(x', y') = (fx/z, fy/z) \tag{5.9}$$

Since this representation does not have simple derivatives of x' and y' with respect to the camera transform parameters, the camera transform is reparameterized to express it in terms of parameters that are related to the camera coordinate system rather than world coordinates. Let D_x and D_y be the position of the origin of the world coordinate projected in the image plane and D_z be the z component of the origin of the world coordinate expressed in camera coordinate system. The new transform is expressed as:

$$(x, y, z) = \mathbf{R}\,\mathbf{p} \tag{5.10}$$

$$\begin{aligned} (x', y') &= (fx/(z+D_z) + D_x ,\ \ fy/(z+D_z) + D_y) \\ &= (fxc + D_x,\ fyc + D_y) \end{aligned} \tag{5.11}$$

where $c = 1 / (z + D_z)$

Here x component of image projection, x', becomes the sum of the projection of the origin (D_x) and the offset projection ($fx/(z+D_z)$) of the (x,y,z) relative to the origin. Y component image projection, y', is obtained in the same way. The camera position **t** and (D_x, D_y, D_z) are related as:

$$\mathbf{t} = \mathbf{R}^{-1}\ (-(D_x(z+D_z))/f,\ -(D_y(z+D_z))/f,\ -D_z\ )^T \qquad (5.12)$$

The derivatives of x, y, and z (hence of x' and y') with respect to rotation parameters (θ,ϕ,ψ) can be expressed in very simple forms. For example, the derivatives of a point (x,y) with respect to a conterclockwise rotation ψ about z axis are ((x1,y1,z1) is rotated to (x,y,z1))

$$\begin{aligned}\frac{\partial x}{\partial \psi} &= \frac{\partial (x_1\cos\psi - y_1\sin\psi)}{\partial \psi} = -x_1\sin\psi - y_1\cos\psi = -y \\ \frac{\partial y}{\partial \psi} &= \frac{\partial (x_1\sin\psi + y_1\cos\psi)}{\partial \psi} = x_1\cos\psi - y_1\sin\psi = x\end{aligned} \qquad (5.13)$$

The other derivatives of a point (x,y) with respect to rotation θ,ϕ are computed similarly.

The partial derivatives of image point (x', y') with respect to each of the new camera parameters (D_x, D_y, D_z, θ, ϕ, ψ) can be computed. For example,

$$x' = \frac{fx}{z + D_z} + D_x \qquad (5.14)$$

$$\frac{\partial x'}{\partial D_x} = 1\ ,\quad \frac{\partial x'}{\partial D_y} = 0\ ,\quad \frac{\partial x'}{\partial D_z} = \frac{-fx}{(z + D_z)^2} \qquad (5.15)$$

$$\begin{aligned}\frac{\partial x'}{\partial \theta} &= -\frac{fx}{(z + D_z)^2}\frac{\partial z}{\partial \theta} = -fc^2xy \\ \frac{\partial x'}{\partial \phi} &= \frac{f}{z + D_z}\frac{\partial x}{\partial \phi} - \frac{fx}{(z + D_z)^2}\frac{\partial z}{\partial \phi} = fcz + fc^2x^2 \\ \frac{\partial x'}{\partial \psi} &= \frac{f}{z + D_z}\frac{\partial x}{\partial \psi} = -fcy\end{aligned} \qquad (5.16)$$

y' derivatives can be calculated in a similar way. Since the initial specification of **R** and **t** can be computed from the surface information contained in surface cluster graph, the Newton-Raphson method is now carried out by correcting errors in x' and y'. This is done by calculating the optimum correction rotations $\Delta\theta$, $\Delta\phi$, and $\Delta\psi$ to be made about image axes and position corrections ΔD_x, ΔD_y, ΔD_z. Instead of adding these corrections to underlying parameters (θ, ϕ, ψ) of **R**, rotations of the given magnitudes ($\mathbf{R}_{\Delta\theta}$, $\mathbf{R}_{\Delta\phi}$, $\mathbf{R}_{\Delta\psi}$) about their respective coordinate axes are computed and new rotation transformation is composed with **R**.

Given the partial derivatives of x' and y', it is easy to achieve convergence. For each point in the model which should match against some corresponding point in the image, the camera transform of the model point is first calculated and the errors in its x and y components ($\Delta x'$, $\Delta y'$) are measured when compared to the given image point. Then equations which express the errors as the sum of the products of its partial derivative times the error correction values are created:

$$\frac{\partial x'}{\partial D_x}\Delta D_x + \frac{\partial x'}{\partial D_y}\Delta D_y + \frac{\partial x'}{\partial D_z}\Delta D_z + \frac{\partial x'}{\partial \theta}\Delta\theta + \frac{\partial x'}{\partial \phi}\Delta\phi + \frac{\partial x'}{\partial \psi}\Delta\psi = \Delta x' \qquad (5.17)$$

$$\frac{\partial x'}{\partial D_x}\Delta D_x + \frac{\partial x'}{\partial D_y}\Delta D_y + \frac{\partial x'}{\partial D_z}\Delta D_z + \frac{\partial x'}{\partial \theta}\Delta\theta + \frac{\partial x'}{\partial \phi}\Delta\phi + \frac{\partial x'}{\partial \psi}\Delta\psi = \Delta y' \qquad (5.18)$$

So for each point correspondence two equations are derived. From three point correspondences, six equations are derived, which can be solved for all six camera model corrections. The

error equations (5.17) and (5.18) can be expressed as a matrix equation

$$\mathbf{A}\ \mathbf{P} = \mathbf{E} \tag{5.19}$$

where $\mathbf{A}$ is the derivative matrix which is computed from (5.15) and (5.16), $\mathbf{P}$ is the vector of unknown corrections, and $\mathbf{E}$ is the column matrix of error terms. Once the correction values of camera parameters ($\mathbf{P}$) are obtained, camera parameters are updataed for the next cycle of convergence.

$$\begin{aligned} D_x^{N+1} &= D_x^N + \Delta D_x \\ D_y^{N+1} &= D_y^N + \Delta D_y \\ D_z^{N+1} &= D_z^N + \Delta D_z \\ \mathbf{R}^{N+1} &= \mathbf{R}_{\Delta\psi}\ \mathbf{R}_{\Delta\phi}\ \mathbf{R}_{\Delta\theta}\ \mathbf{R}^N \end{aligned} \tag{5.20}$$

Here superscript refers to iteration number. Iteration terminates when the average error is within tolerance.

In most applications of this method we will be given more correspondences between model and image than are strictly necessary, and we will want to perform some kind of best fit. In this case we can perform a least-squares fit of the errors simply by solving the normal equation.

$$\mathbf{A}^T\ \mathbf{A}\ \mathbf{P} = \mathbf{A}^T\ \mathbf{E} \tag{5.21}$$

where $\mathbf{A}^T\mathbf{A}$ is square and has the correct dimension for the vector $\mathbf{P}$.

5.4 Object Recognition

By recognition we mean that identity, location, and orientation of objects have been determined. Low level image analysis up to now gave rise to a description of the image by surface patches or image curve features. A description of the image or scene thus is given in the form of a relational structure in which the nodes correspond to features, surface patches, or objects, labeled by lists of their property values (shape, surface types) and arcs correspond to the relationships between these features.

Recognition is a search process establishing the best match between a model and an image. This process can be split into two subproblems: the problem of identifying corresponding surface patches or points in the model and the image, and the problem of computing the degree of match between these two sets of corresponding surface patches or points. Surface information, as extracted in Chapter 4, provides useful information for solving the two problems.

5.4.1 Candidate Model Generation

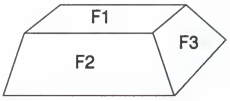
One important and difficult task for a general model-based vision system is selecting the correct model. Model-based vision is computationally intractable without reducing

the large set of objects that potentially explain a set of image features to a few serious candidates that require more detailed analysis. Suppose we have m model features and i image features. The possible pairings are $m \times i$. If we want to identify an object of p features, the total number of matching required is p combinations out of $m \times i$ ($_{m \times i}C_p$). If point features (such as vertex or corner) are used, the problem is extremely difficult. If lines or curves are used, we can achieve a minor improvement. If surface information is used, the improvement is substantial because the number of surfaces are much smaller than lines or point features, and the surface feature itself suggests possible objects. The problem is too large to undertake a model-directed comparison of every known object from every viewpoint. Model invocation is the result of the directed convergence of clues that suggest identities for explaining image features. Model invocation is computed through associations or relationships with other objects. The effect is one of suggestion, rather than confirmation.

In order to reduce the search space, we must have a domain independent grouping of image features. But too much grouping may degrade the robustness of the features, since we might misgroup them without domain specific knowledge. Little structuring is general enough to be used in noisy situations but we have to pay for the associated combinatorial problem.

Model invocation starts with a surface cluster graph generated from the integration module. From a surface cluster

graph we first identify all the unoccluded, completely visible surfaces. A surface is completely visible if its enclosing edge segments do not contain the shaft of the T junction as one of its boundary. Each completely visible surface is searched through the surface classification network and is associated with a list of possible objects which have that surface as one of its bounding surfaces. These lists form the first level of ranking. Pair-wise logical disjunctions between the lists form the second level of ranking. This logical disjunction of lists of objects continues until we get the logical disjunction of all the lists of objects associated with each completely visible surface. Since objects at higher levels are also contained in lower levels, the objects which belong to a high level are eliminated from a lower level lists simply by subtracting those objects. The meaning of each level of ranking is that, in highest ranked list, we have objects which contain all the completely visible surfaces as their bounding surfaces. The next level indicates that the objects in that level have all the visible surfaces except one as their bounding surfaces. This interpretation applies to the lowest level where only one visible surface is associated with the objects which have that surface as one of the bounding surfaces. A list of objects linked with each surface results in a first level pruning of the candidate models. Figure 5.5 shows an example of a ranked candidate list.



Lowest ranked candidate objects

```
F1  --->  ( obj1  obj3  obj5  obj9  obj11  obj13 )
F2  --->  ( obj3  obj5  obj6  obj8  obj11 )
F3  --->  ( obj1  obj3  obj6  obj9  obj11  obj13 )
```

Second highest ranked candidate objects

```
F1 ∩ F2  --->  ( obj3  obj5  obj11 )
F1 ∩ F3  --->  ( obj1  obj3  obj 9  obj11  obj13 )
F2 ∩ F3  --->  ( obj3  obj6  obj11 )
```

Highest ranked candidate objects

```
F1 ∩ F2 ∩ F3  --->  ( obj3  obj11 )
```

Figure 5.5 An example of candidate ranking. Objects which have all three surfaces as their bounding surfaces are ranked highest.

Now, the candidate model list is tested one by one from the highest ranked list (objects which have all the surfaces in the image as their bounding surface) to the lowest ranked list by computing the viewing transformation and verifying that the transformed object model conforms with the image description. Before matching a candidate model with the image features, another screening process to eliminate an impossible candidate model fast is done by first examining the adjacency and geometric constraints of visible surfaces for each model. Let P_s = (s1, s2) be a surface pair in the surface cluster graph, and P_m = (m1, m2) be a surface pair in the model. In order to be the corresponding pair of P_s, P_m must satisfy the following conditions.

$$| \text{Ang}_{s1s2} - \text{Ang}_{m1m2} | \; < \; 15 \text{ degree}$$

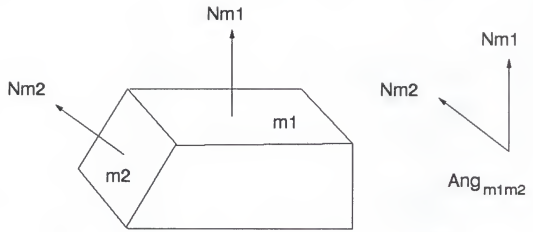
$$| \text{Area}_{s1}/\text{Area}_{s2} - \text{Area}_{m1}/\text{Area}_{m2} | \; < \; 0.2 * (\text{Area}_{m1}/\text{Area}_{m2})$$

Ang_{s1s2} is the angle between the surface normals of s1 and s2 in the image, and Ang_{m1m2} is the angle between the surface normals of m1 and m2 in the model. Area_{s1}, Area_{s2}, Area_{m1}, and Area_{m2} are areas of s1, s2, m1, and m2, respectively. Surface normals are viewpoint variant features, but the angle between two surface normals is a view-invariant feature. If a surface normal is obtained accurately, Ang_{s1s2} is the same as Ang_{m1m2}, no matter what viewing directions we look at the object. Since noise is inevitable in image, the angle condition accommodates

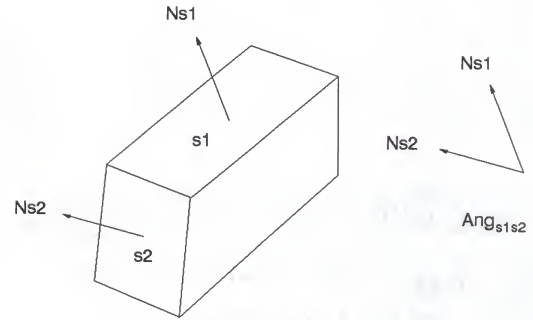
any possible surface normal computation error of the photometric stereo method. The surface area observed in the image is dependent on the orientation of the surface and on the distance with respect to the viewing direction. Figure 5.6 shows the angle-area relationship between a model and its corresponding image. We showed in section 4.5.2 that we can recover a surface area up to a scale factor for planar regions. But area ratio is independent of scale factor, hence it is a more useful feature. Considering segmentation error and surface normal error, an area ratio in an image within 20 % of a model area ratio is accepted as a valid surface pair. These conditions are based on the rotation and translation invariant properties of the objects, and the angle condition is valid even under the existence of occlusions. A candidate model is discarded if it does not satisfy the above conditions.

5.4.2 Computation of Initial Viewpoint Parameters

For each candidate model, a rotation transform is computed first by associating the primary surface and one of its neighbor in the image with the corresponding surfaces in a candidate model. We discussed in section 5.3.1 that a rotation transform can be composed by three elementary rotations about x, y, and z axes. A rotation transform can be represented another way by a rotation of angle θ about an arbitrary unit vector $\underline{k}$ (axis of revolution) located at the



(a) A model



(b) An image of the model

Figure 5.6 Angle relationship between a model and the image

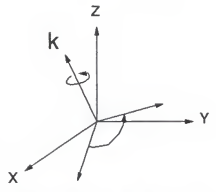
origin. Let k_x, k_y, and k_z represent the three components of the axis $\underline{k}$, then rotation **Rot** about axis $\underline{k}$ can be expressed as follows [Paul 1984].

$$\mathbf{Rot}(k,\theta) = \begin{pmatrix} k_xk_xv+c & k_yk_xv-k_zs & k_zk_xv+k_ys \\ k_xk_yv+k_zs & k_yk_yv+c & k_zk_yv-k_xs \\ k_xk_zv-k_ys & k_yk_zv+k_xs & k_zk_zv+c \end{pmatrix} \qquad (5.22)$$

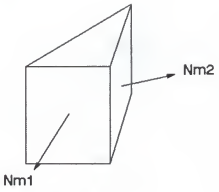
where $s = \sin\theta$, $c = \cos\theta$, and $v = 1 - \cos\theta$.

Figure 5.7(a) shows the rotation transform about an arbitrary axis $\underline{k}$. We first find the rotation transform aligning model surface normal $\underline{N}_{m1}$ with the corresponding image surface normal $\underline{N}_{d1}$ (Figure 5.7(d) to Figure 5.7(e)). The axis of rotation is given as $\underline{N}_{m1} \times \underline{N}_{d1}$. The rotation **Rot1** between a model surface normal $\underline{N}_{m1}$ and its corresponding image surface normal $\underline{N}_{d1}$ is then given as (Figure 5.7(b)):

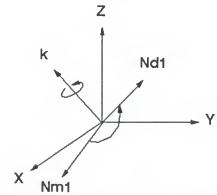
Rot1($\underline{k1}$,θ) where $\underline{k1}=\underline{N}_{m1} \times \underline{N}_{d1}$ and θ = arccos ($\underline{N}_{m1} \cdot \underline{N}_{d1}$). Since this rotation transform has one degree of freedom about the N_{d1} axis, the final rotation transform is obtained by aligning $\underline{N}_{m2}$ with $\underline{N}_{d2}$ (Figure 5.7(e) to Figure 5.7(f)). The rotation about N_{d1} is given as **Rot2**($\underline{k2}$,ϕ) where $\underline{k2}$ is $\underline{N}_{d1}$, ϕ is the angle between the projections of $\underline{N}_{d2}$ and R2•$\underline{N}_{m2}$ onto the plane perpendicular to $\underline{N}_{d1}$ (Figure 5.7(c)). The initial rotation transform is given by **Rot2** • **Rot1**. This transform aligns two model surface normals with the corresponding two image surface normals.



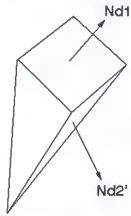
(a) rotation about arbitrary axis k



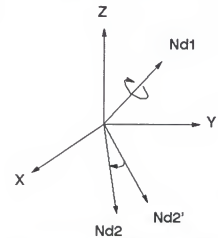
(d) a model



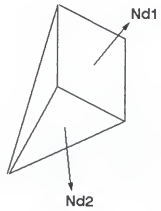
(b) rotation aligning Nm1 and Nd1



(e) projection after Nm1 and Nd1 alignment



(c) rotation aligning Nd2' and Nd2



(f) an image

Figure 5.7 Initial rotation transform computation

Initial position estimates are provided by the image location of the primary surface. The distance between the camera and an object in the image is estimated by comparing the size of the rotation-transformed primary surface with the corresponding surface size in the image.

5.4.3 Verification

Since we are using an iterative method for computing the transformation, the two primary surfaces in the surface cluster graph is used until they are reasonably matched by a candidate model. The initial points used for verification are the vertices of the common edge of two primary surfaces and two additional vertices adjacent to the common edge vertices.

The iteration cycle of refining the view transform computation starts by projecting a candidate model into the image plane, using the estimated view transform. The corresponding vertices are compared to measure the errors, then a correction vector of six parameters is computed to update the current parameters. The projection, error measure, and parameter update cycle repeat. Since the iterative method converges very fast, we can determine possible matching within several cycles by observing the error behavior. Once the error is within the specified error bound, we include other vertices for verification. When a vertex is not available due to occlusion, a line error between an image line and a projected model line is used for error measurement. A line

error is defined as the maximum distance of endpoints of one line from the other line.

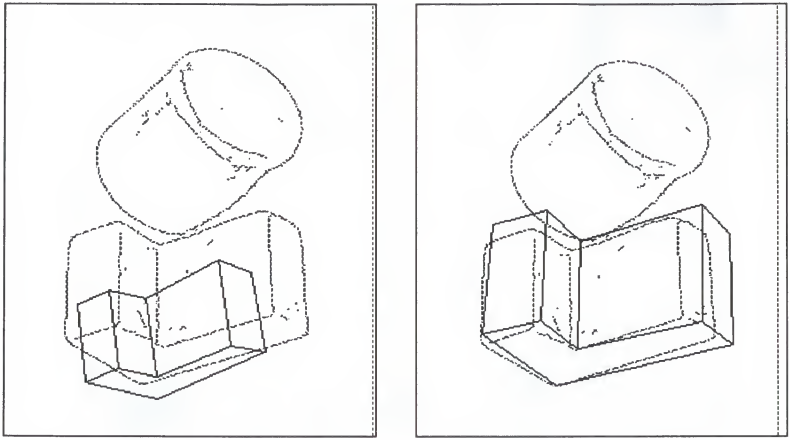
The projection of the object by the transformation requires a visibility check of each surface. Currently, we implemented a rudimentary visibility calculation. A surface is visible if the transformed normal is visible. A vertex and an edge are visible if the corresponding surfaces are visible.

First, all the visible vertices and edges in a surface cluster graph are verified. Once this is done, we can now predict other visible surfaces by projecting the candidate model, using visibility calculation. The verified surfaces are then excluded from the list of remaining surfaces to be examined further by the recognition procedure. After one object recognition is completed, the next surface cluster graph in the remaining list is examined by the same procedure. This process continues until all the objects in the scene are recognized.

5.5 Experimental Results

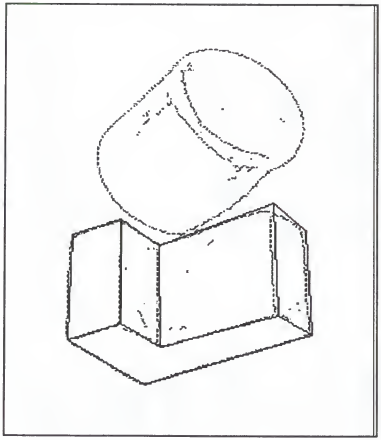
The recognition algorithm was implemented and tested for a few scenes. Figure 5.8 shows the recognition cycle of an image of a cylinder and a stair. The initial transformation was estimated using the two largest surfaces. We included the camera focal length as an unknown variable. Recognition took longer than in the other two experiments because the focal length was also unknown. Figure 5.9 is the recognition cycle

of an image of a stair and a trough. The trough was recognized first because it has the largest top area. Initial estimate of distance was large. At the next iteration it overshot a little, then immediately converged. Figure 5.10 shows the recognition of a stair in four iterations of the same image. Table 5.1, 5.2, and 5.3 are the view transform parameters of each recognition cycle. x_t, y_t, and z_t are the the components of the translation vector between the object centered coordinate system and the image coordinate system. θ, ϕ, and ψ are rotation angles with respect to X, Y, and Z coordinates. These parameters are defined in section 5.3.1.

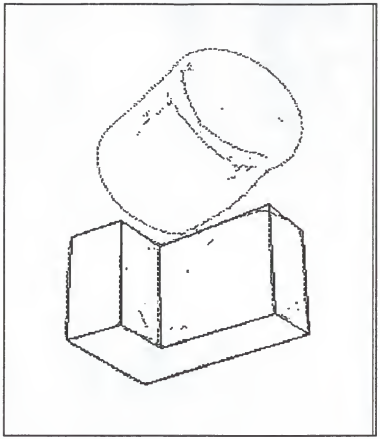


(a) (b)

Figure 5.8 Stair Recognition (a) iteration no.=1 (b) iteration no.=3
(c) iteration no.=5 (d) iteration no.=6

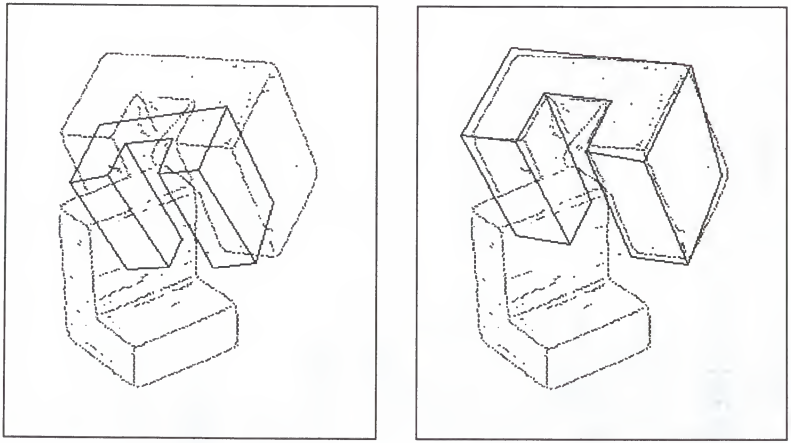


(c)



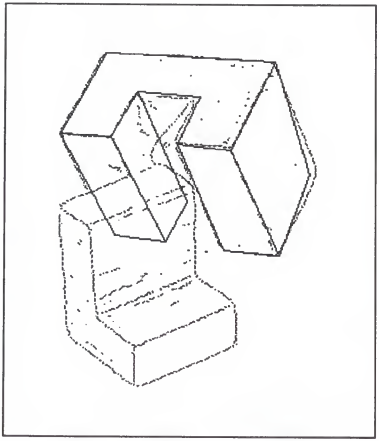
(d)

Figure 5.8--continued

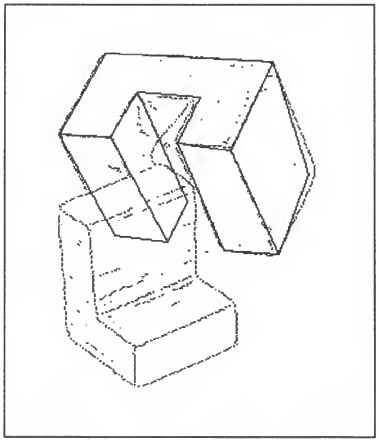


(a) (b)

Figure 5.9 Trough recognition (a) iteration no.=1 (b) iteration no.=2
(c) iteration no.=3 (d) iteration no.=4

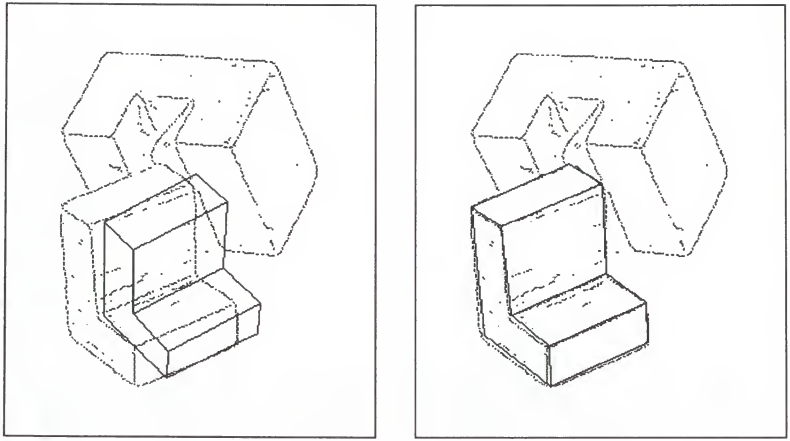


(c)



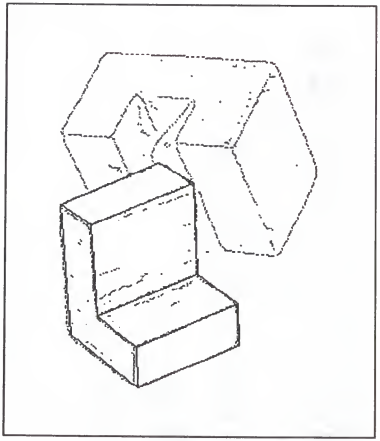
(d)

Figure 5.9--continued

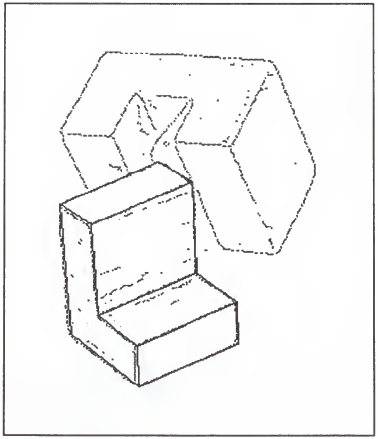


(a) (b)

Figure 5.10 Stair recognition (a) iteration no.=1 (b) iteration no.=2
(c) iteration no.=3 (d) iteration no.=4



(c)



(d)

Figure 5.10--continued

Table 5.1 View Parameters for the Stair

	iteration=1	iteration=3	iteration=5	iteration=6
x_t (cm)	-35.6	-16.3	-33.8	-34.0
y_t (cm)	-29.8	-13.4	-26.4	-26.9
z_t (cm)	100.4	63.9	68.27	69.7
θ (deg)	21.3	25.6	26.0	26.1
ϕ (deg)	135.9	146.8	144.5	144.5
ψ (deg)	173.0	-177.7	179.7	179.5
f	715.7602	813.8250	992.0088	993.5185

Table 5.2 View Parameters for the Trough

	iteration=1	iteration=2	iteration=3	iteration=4
x_t (cm)	21.4	16.4	16.0	15.7
y_t (cm)	25.9	17.2	19.7	19.9
z_t (cm)	84.0	60.9	67.6	68.4
θ (deg)	-42.9	-31.9	-35.0	-35.5
ϕ (deg)	56.8	69.1	65.8	65.3
ψ (deg)	161.9	158.7	159.0	158.9

Table 5.3 View Parameters for the Stair

	iteration=1	iteration=2	iteration=3	iteration=4
x_t (cm)	3.5	-4.9	-3.6	-3.6
y_t (cm)	97.8	70.9	71.7	71.7
z_t (cm)	27.7	23.7	23.9	23.9
θ (deg)	-28.7	-41.4	-39.8	-39.8
ϕ (deg)	117.0	127.9	126.7	126.7
ψ (deg)	-136.0	-126.7	-128.1	-128.1

CHAPTER 6
CONCLUSION

In this dissertation, we have developed an integrated segmentation approach to the three-dimensional object recognition problem based on the edge and surface orientation maps. Unlike gray level images, the surface orientation map provides rich and explicit three-dimensional information of a scene, namely a surface normal vector at each pixel location of the scene. This rich three-dimensional information must first be organized into meaningful structures (e.g., surface patch description) to understand the scene.

Segmentation of an image, or partitioning of the image into connected regions, each of which has different physical properties, is the most difficult and crucial step toward a high-level symbolic description of the image, which is essential for three-dimensional object recognition. There have been two approaches to the image segmentation problem: an edge-based approach and a region-based approach. The edge-based approach invariably produces spurious edges as well as real edges and misses some real edges, thus region boundaries are rarely closed. On the other hand, segmentation results from the region-based approach depend on the seed regions chosen and the region growing methods. The region-based

approach does produce disjoint divisions of the given image but often either oversegments or undersegments, hence some region boundaries do not correspond to the physical boundaries of the objects in the scene. Another major drawback of the region-based approach is its enormous computational burden for the region merging and splitting operations.

In this dissertation, we developed an integrated approach to the image segmentation problem, which compensates for the deficiencies of using either approach. The following contributions are achieved by taking the integrated approach.

(1) Reliable and coherent segmentation

Our main contribution is an improved segmentation by fusing the edge detection process and a surface-based segmentation process into a much more reliable and coherent surface patch description to facilitate object recognition. During integration, consistent segmentation cues from different channels reinforce one another, and the contradictory cues from noise or segmentation error are canceled by examining a global consistency. By registering the edge detection output with the surface-based segmentation, explicit edge types (e.g., convex, concave) can be assigned to each edge segment. Broken edge segments are connected and junctions are more reliably found . The prediction of possibly missing edges is possible by using the junction dictionary. Region splitting is facilitated by locating a potential

region boundary. The integrated segmentation result is an explicit and rich description of edges and surfaces of objects in the scene, which facilitates object recognition.

(2) Computational efficiency

An increase in speed is achieved by employing a divide and conquer strategy during region analysis. The focus of attention is given to the large regions enclosed by edge segments. Regions which are adjacent to a large region and inside the boundary of a larger region are very likely to be the same surface type. These regions are usually segmentation artifacts. Region merging is first tried on those regions which are very likely to be one region.

(3) High Curvature regions for surface based segmentation

We introduced high curvature regions to facilitate the segmentation by preventing the same surface types with different surface orientation from being labeled as the same region. In addition, high curvature regions make the interpretation of registered edge detection output and surface property based segmentation output easy.

(4) Surface classification network and pre-filtering

We proposed to use a surface classification network, where each surface type is associated with a list of objects which have that surface as one of the bounding

surfaces, to prune the search space of objects in the hypotheses generation phase. Hypotheses are ranked and adjacency and geometric constraints are used to further prune each hypothesis before the verification.

Compared to the other uses of a surface orientation map (e.g., the Extended Gaussian Image approach), the integrated approach provides much more reliable and coherent description of surface patches and their relationships in the image.

The most difficult problem in the segmentation of the surface orientation map is that shadow regions in the scene cannot generate surface orientation information, causing shadow boundaries. Surface based segmentation has no way of knowing this effect. If we use a three edge information from three different illumination directions, which must be used in the photometric stereo method, and the knowledge of illumination directions, detection of shadow boundaries and regions will improve. Research in this direction could provide a solution to the shadow problem.

Another direction of research is improving the reliability of the photometric stereo technique. Currently, the photometric stereo method works for objects of smooth surfaces. Further research is required to reliably obtain a surface orientation map for objects with textured or non-smooth surfaces. In the current implementation, verification of the hypothesized object is tried only for polyhedral objects. General verification requires rendering capability

of general curved objects within an acceptable time requirement. Progress in the computer graphics field will provide a solution to this problem.

REFERENCES

Agin, G.J. and Binford, T.O., "Computer Description of Curved Objects," Proc. of 3rd International Joint Conf. on Artificial Intelligence, Stanford, CA, 1973, pp. 629-640.

Ahn, H.Y. and Tou, J.T., "Segmentation via Fusion of Edge and Needle Map," Proc. of SPIE Conf. on Applications of Artificial Intelligence IX, Orlando, FL, 1991.

Asada, H. and Brady, M., "The Curvature Primal Sketch," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, No. 1, 1986, pp. 2-14.

Ayache, N., "A Model-based Vision System to Identify and Locate Partially Visible Industrial Parts," Proc. of the IEEE Computer Vision and Pattern Recognition, Washington, D.C., 1983, pp. 492-494.

Ballard, D.H. and Brown, C.M., Computer Vision, Prentice-Hall, Englewood Cliffs, NJ., 1982.

Barnard, S.T., "Interpreting Perspective Images," Artificial Intelligence, vol. 21, 1983, pp. 435-462.

Barrow, H.G. and Popplestone, R.J., "Relational Descriptions in Picture Processing," In Machine Intelligence VI, B. Meltzer and D. Michie (eds.), American Elsevier, New York, 1971.

Barrow, H.G. and Tenenbaum, J.M., "Interpreting Line Drawings as Three-Dimensional Surfaces," Artificial Intelligence, vol. 17, 1981, pp. 75-116.

Barrow, H.G. and Tenenbaum, J.M., "Computational Vision," Proc. of IEEE, vol. 69, No. 5, July 1983, pp. 572-595.

Bertero, M., Poggio, T.A. and Torre, V., "Ill-Posed Problems in Early Vision," Proc. of the IEEE, vol. 76, No. 8, August 1988, pp. 869-889.

Berzins, V., "Accuracy of Laplacian Edge Detectors," Computer Vision, Graphics, Image Processing, vol. 27, 1984, pp. 195-210.

Besl, P.J. and Jain, R.C., "Three-dimensional Object Recognition," ACM Computing Surveys vol. 17, No. 1, March 1985, pp. 75-145.

Besl, P.J. and Jain, R.C., "Invariant Surface Characteristics for 3-D Object Recognition in Range Images," Computer Vision, Graphics, Image Processing, vol. 33, 1986, pp. 33-80.

Besl, P.J. and Jain, R.C., "Segmentation via Variable-Order Surface Fitting," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, No. 2, March 1988, pp.167-192.

Bhanu, B., "Representation and Shape Matching of 3-D Objects," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, No. 3, May 1984, pp. 340-350.

Bolles, R.C. and Cain, R.A., "Recognizing and Locating Partially Visible Objects: The Local Feature-Focus Method," Int. J. of Robotics Research, vol. 1, No. 3, Fall 1982, pp. 637-643.

Bolles, R.C. and Horaud, P., "3DPO: A Three-Dimensional Part Orientation System," Int. J. of Robotics Research, vol. 5, No. 3, Fall 1986, pp. 3-26

Brady, M., "Computational Approaches to Image Understanding," ACM Computing Surveys, vol. 14, No. 1, March 1982, pp. 3-71.

Brady, M., Yuille, A., "An Extremum Principles for Shape from Contours," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, No. 3, 1984, PP. 288-301

Brice, C. and Fennema, C., "Scene Analysis using Regions", Artificial Intelligence, vol. 1, 1973, pp. 205-226.

Brooks, R.A., "Symbolic Reasoning among 3-D Models and 2-D Images," Artificial Intelligence, vol. 17, 1981, pp. 285-348.

Brooks, R.A., "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, No. 2, March 1983, pp. 140-150.

Canny, J., "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, No. 6, November 1986, pp. 679-698.

Chakravarty, I. and Freeman, H., "Characteristic Views as a Basis for Three-Dimensional Object Recognition," Proc. of SPIE Conf. on Robot Vision, Arlinton, VA., vol. 336, May 1978, pp. 37-45.

Chen, P.C. and Pavlidis, T., "Image Segmentation Problem as an Estimation Problem," Computer Graphics, Image Processing, vol. 12, 1980, pp. 153-172.

Chin, R.T. and Dyer, C.R., "Model-Based Recognition in Robotic Vision," ACM Computing Surveys, vol. 18, No. 1, March 1986, pp. 67-108.

Chow, K.C. and Kaneco, T., "Boundary Detection of Radiographic Images by a Thresholding Method," in Frontiers of Pattern Recognition, S. Watanabe, Ed., Academic Press, New York, 1972, pp. 61-82.

Davis, L.S., "A Survey of Edge Detection Techniques", Computer Graphics and Image Processing, vol. 4., 1975, pp.248-270

Dhome, M. and Kasvand, T., "Polyhedra Recognition by Hypothesis Accumulation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, May 1988, pp. 429-438.

Do Carmo, M.P., Differential Geometry of Curves and Surfaces, Prentice-Hall, Englewood Cliffs, NJ, 1976.

Duda, R.O. and Hart, P.E., "The Use of Hough Transform to Detect Lines and Curves in Pictures," Communications of the ACM, vol. 15., 1972, pp. 11-15.

Duda, R.O. and Hart, P.E., Pattern Classification and Scene Analysis, John Wiley & Sons, Inc., New York, 1973.

Duda, R.O., Nitzan, D., and Barrett, P., "Use of Range and Reflectance Data to Find Planar Surface Regions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, No. 3, July 1979, pp. 254-271.

Faugeras, O.D. and Hebert, M., "The Representation, Recognition and Locating of 3-D Objects," Int. J. of Robotics Research, vol. 5, No. 3, Fall 1986, pp. 27-52.

Fischler, M.A. and Bolles, R.C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," CACM vol 24, No. 6, 1981, pp. 381-395.

Fischler, M.A. and Bolles, R.C., "Perceptual Organization and Curve Partitioning," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, No. 1, November 1986, pp. 100-105.

Foley, J.D. and Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison-Wesley, Reading, MA, 1982.

Goad, C., "Special-Purpose Automatic Programming for 3-D Model-Based Vision," Proc. of DARPA Image Understanding Workshop, Arlington, VA., June 1983, pp. 94-104.

Grimson, W.E.L. and Lozano-Perez, T., "Model-Based Recognition and Localization from Sparse Range or Tactile Data," Int. J. of Robotics Research, vol. 3, No. 3, Fall 1984, pp. 3-35.

Grimson, W.E.L. and Lozano-Perez, T., "Localizing Overlapping Parts by Searching the Interpretation Tree," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, No. 4, July 1987, pp. 469-482.

Haralick, R.M., "Digital Step Edges from Zero-Crossings of Second-Directional Derivatives," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, No. 1, January 1984, pp. 58-68.

Haralick, R.M. and Shapiro, L.G., "Image Segmentation Techniques," Computer Vision, Graphics, Image processing, vol. 29., 1985, pp. 100-132.

Haralick, R.M. and Watson, L.T., and Laffey, T.J., "The Topographic Primal Sketch," Int. J. of Robotics Research, vol. 2, No. 1, Spring 1983, pp. 50-72.

Henderson, T.C., "Efficient 3-D Object Representations for Industrial Vision Systems," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 3, No. 6, November 1983, pp. 609-617.

Hoffman, D.D. and Richards, W.A., "Parts of Recognition," Cognition, vol. 18, 1985, pp. 65-96.

Horaud, R., "New Methods for Matching 3-D Objects with Single Perspective," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, May 1987, pp. 401-412.

Horn, B.K.P., "Understanding Image Intensities", Artificial Intelligence, vol. 8, No. 2, April 1977, pp. 201-231.

Horn, B.K.P., "Extended Gaussian Images," Proc. of the IEEE, vol. 72, December 1984, pp. 1656-1678.

Horn, B.K.P. and Ikeuchi, K., "The Mechanical Manipulation of Randomly Oriented Parts,", Scientific America, vol. 251, No. 2, August 1984, pp. 100-111.

Horowitz, S.L. and Pavlidis, T., "Picture Segmentation by a Tree Traversal Algorithm," J. of Assoc. of Computing Machinery, vol. 23, 1976, pp. 368-388.

Hsiung, C.C., A First Course in Differential Geometry, Wiley-Interscience, New York, 1981.

Huffman, D.A., "Impossible Objects as Nonsense Sentences," Machine Intelligence, vol. 6, 1971, pp.295-323.

Huttenlocher, D.P. and Ullman, S., "Object Recognition using Alignment," Proc. First Int. Conf. Computer Vision and Pattern Recognition, London, June 1987, pp. 594-601.

Ikeuchi, K., "Recognition of 3-D Objects Using the Extended Gaussian Image," Proc. 7th Int. Joint Conf. on Artificial Intelligence, Vancouver, B.C., Canada, August 1981, pp. 918-920.

Ikeuchi, K. and Horn, B.K.P., "Numerical Shape from Shading and Occluding Boundaries," Artificial Intelligence, vol. 17, August 1981, pp. 141-184.

Kanade, T., "Survey: Region Segmentation: Signal vs. Semantics," Computer Graphics and Image Processing, vol. 13, 1980, pp. 279-297.

Levine, M.D. and Nazif, A.M., " Low Level Image Segmentation: An Expert System," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, No. 5, September 1984, pp. 555-577.

Lowe, D.G., "Three-Dimensional Object Recognition from Single Two-Dimensional Images," Artificial Intelligence, vol. 31, 1987, pp. 355-395.

Malik, J., "Interpreting Line Drawings of Curved Objects," Int. J. of Computer Vision, vol. 1, 1987, pp. 73-103

Marr, D., Vision, Freeman, New York, 1982.

Marr, D. and Hildreth, E.C., "Theory of Edge Detection," Proc. of Royal Society of London, B 207, 1980, pp. 187-217.

Marr, D. and Nishihara, K., "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes," Proc. of Royal Society of London, B 200, 1977, pp. 269-294.

Marr, D. and Poggio, T., "A Theory of Human Stereo Vision," Proc. of Royal Society of London, B 204, 1979, pp. 301-328.

McKeown, D.M., Harvey, W.A., Mcdermott, J., "Rule-based Interpretation of Aerial Imagery," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 7, No. 5, 1985, pp. 570-585.

Medioni, G. and Nevatia, R., "Description of 3-D Surfaces Using Curvature Properties," Proc. of Image Understanding Workshop, New Orleans, LA, 1984, pp 291-299

Mulgaonkar, P.G. and Shapiro, L.G., "Hypothesis-Based Geometric Reasoning about Perspective Images," IEEE 3rd Workshop on Computer Vision, 1985, pp. 11-18.

Nevatia, R. and Babu, K.R., "Linear Feature Extraction and Description," Computer Graphics and Image Processing, vol. 13, 1980, pp. 257-269.

Nevatia, R. and Binford, T.O., "Description and Recognition of Complex-Curved Objects," Artificial Intelligence, vol. 8, 1977, pp. 77-98.

Ohlander, R., Analysis of Natural Scenes, Ph.D. Dissertation, Carnegie-Mellon University, Pittsburgh, PA, 1975.

Oshima, M. and Shirai, Y., "Object Recognition Using Three-Dimensional Information," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 3, No. 4, July 1983, pp. 353-361.

Park, J.S. and Tou, J.T., "Highlight Separation and Surface Orientations for 3-D Specular Object," Proc. of 10th Int. Conf. Pattern Recognition, Atlantic City, NJ, June 1980, pp. 331-335

Paul, R.P., Robot Manipulators: Mathematics, Programming, and Control, The MIT Press, Inc., Cambridge, MA, 1984

Pavlidis, T., Algorithms for Graphics and Image Processing, Computer Science Press, Inc., Rockville, MD, 1982.

Pavlidis, T., Structural Pattern Recognition, Springer-Verlag, Berlin, 1977.

Pavlidis, T. and Liow, Y., "Integrating Region Growing and Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, No. 3, 1990, pp. 225-233.

Pentland, A.P., "Perceptual Organization and the Representation of Natural Form," Artificial Intelligence, vol. 28, 1986, pp. 293-331.

Prewitt, J.M.S., "Object Enhancement and Extraction," in Picture Processing and Pschopictorics, B. Lipkin and A. Rosenfeld (eds.), Academic Press, New York, 1970, pp. 75-149.

Requicha, A.A., "Representations for Rigid Solids: Theory, Method, and Systems," ACM Computing Surveys, vol. 12, No. 4, December 1980, pp. 437-464.

Roberts, L.G., "Machine Perception of Three-Dimensional Solids," Optical and Electro-Optical Information Processing, J.T. Tippett et al. (eds.), MIT Press, Cambridge, MA, 1963, pp. 159-197.

Rosenfeld, A. and Kak, A., Digital Picture Processing, vols. 1 and 2, Academic Press, New York, 1982.

Strat, T.M., "Recovering the Camera Parameters from a Transformation Matrix," Proc. DARPA Image Understanding Workshop, New Orleans, LA, 1984, pp. 264-271.

Sugihara, K., "Range-data Analysis Guided by Junction Dictionary," Artificial Intelligence, vol. 12, No. 1, 1979, pp. 41-69.

Tech Tran Consultants Inc., Machine Vision Systems, McGraw-Hill, New York, 1985.

Tou, J.T., "Knowledge-Based Pattern Recognition and Image Interpretation," Proc. of 15th Southeastern Symposium on System Theory, Huntsville, Albama, 1983.

Tou, J.T., "VIREC-A Visual Recognition Machine for Industrial Parts," The SME Conf. on Artificial Intelligence in Manufacturing, 1986.

Tou, J.T., "Knowledge-Based Pattern Understanding," The 8th International Conference on Pattern Recognition, Paris, October 27-31, 1986.

Tou, J.T. and Fan, K.C., "Geo-Graphic Code Representation of 3-D Objects from Their Pictorial Drawings," Proc. of SPIE 6th Conference on Applications of Artificial Intelligence VI, April 1988

Tou, J.T. and Gonzalez, R.C., Pattern Recognition Principles, Addison-Wesley Pub. Co., New York, 1974.

Tou, J.T. and Huang, C.L., "Recognition of 3-D Objects via Spatial Understanding of 2-D Images," Proc. of IEEE 2nd Conf. on Artificial Intelligence Application, Miami Beach, FL., Dec. 1985, pp. 641-645.

Tropf, H. and Walter, I., "An ATN Model for 3-D Recognition of Solids in Single Images," Proc. 8th Int. Joint Conf. on Artificial Intelligence, Karlsruhe, W. Germany, 1983, pp. 1094-1098.

Waltz, D.L., "Understanding Line Drawings of Scenes with Shadows," in Psychology of Computer Vision, McGraw-Hill, New York, 1975, pp. 19-91.

Witkin, A.P., "Scale-space Filtering," Proc. 8th Int. Joint Conf. on Artificial Intelligence, Karlsruhe, W. Germany, 1983, August 1983, pp. 1019-1022.

Woodham, R.J., "Photometric Method for Determining Surface Orientation from Multiple Images," Optical Eng., vol. 19, 1980, pp. 139-144.

Yakimovsky, Y., "Boundary and Object Detection in Real World Images," J. of ACM, vol. 23, 1976, pp. 599-618.

Zucker, S., Hummel, R. and Rosenfeld, A., "An Application of Relaxation Labeling to Line and Curve Enhancement," IEEE Transactions on Computers, vol. 26, 1977, pp. 394-403.

BIOGRAPHICAL SKETCH

Hongyoung Ahn was born in Taegu, Korea, on October 8, 1952. He received his bachelor's degree in electronic engineering from the Seoul National University in February, 1975, and his master's degree in electrical engineering from the University of Florida in May, 1986.

From 1975 to 1984, he was with the Agency for Defense Development, Daejeon, Korea, where he was a senior researcher. At the Agency for Defense Development, he was awarded an outstanding achievement award in February, 1979.

In August, 1984, he and his family came to Gainesville to do his Ph.D. work with the Department of Electrical Engineering, the University of Florida. Since 1987, he has been serving as a research assistant at the Center for Information Research, the University of Florida, working in the area of computer vision, image processing, and artificial intelligence.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Julius T. Tou, Chairman
Graduate Research Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Leon W. Couch
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

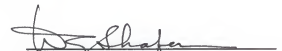
Jose C. Principe
Associate Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yuan-Chieh Chow
Professor of Computer and
Information Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1991



for Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School